SECURITY CLASS

AD-A236 637

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUM 2	ESSION NO. 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Fundamental Theory of Crystal Decomposition	5. TYPE OF REPORT & PERIOD COVERED Final Report 1 Jan 89 - 31 Mar 91
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) A. Barry Kunz	8. CONTRACT OR GRANT NUMBER(s) ONR-N00014-89-J-1601
9. PERFORMING ORGANIZATION NAME AND ADDRESS Michigan Technological University	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of NAval Research Arlington, VA 22217	12. REPORT DATE May 1991
	13. NUMBER OF PAGES 278
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report) unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

DTIC

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release: distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Energetic solids, defects

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

See page 1

91-01671

91 6 10 021

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Final Report

on

FUNDAMENTAL THEORY OF CRYSTAL DECOMPOSITION

A. Barry Kunz
College of Engineering
and
Department of Electrical Engineering
Michigan Technological University
Houghton, MI 49931

May 1991

TABLE OF CONTENTS

Abstract:

A general method for the study of point and extended defects in non-metals has been formulated and a substantial computer program generated to allow the study of such systems in a routine manner. This method requires only a single ansatz; this is the effect of the defect in question is appreciable only in the immediate proximity of the defect. Beyond this region, the influence of the defect may be obtained from a simple response theory, which may be linear but is not required to be so. This response is manifested as a displacement of the ion cores and by the polarization of these atoms. The region including the defect and its immediate vicinity is termed the cluster, while the remainder of the system is termed the environment. The cluster is studied in its entirety using Quantum Physics, whereas the environment is studied using atomistic techniques. [illegible] the potentials are quantum derived. The interaction of the cluster and the environment uses ab initio quantum potentials as well as the rigorous Kunz-Klein localizing potential to partition the system into cluster and environment. The appropriate physical or chemical output is obtained from total energy difference calculations. Initially the environmental relaxation was obtained using the HADES approach developed at Harwell AERE. More recently, the use of Monte Carlo techniques have been developed for this purpose. These have significant advantages, in that finite temperature studies may be included in the studies, and also [illegible] geometries are more easily studied. The extension of the basic method from consideration of point defects to the study of line or planar defects is facilitated here. Correlation corrections are directly incorporated in the quantum calculation by use of Many Body Perturbation Theory (MBPT) methods. The benefit of this [illegible] is demonstrated here in the calculation of the potential energy surface of KrF_2. Other illustrative [illegible] are included as well.

1. Introduction.

Lattice defects in or on crystalline materials, determine many technologically important properties. Reliable computer simulations of such defects are of potential value, and may be expected to contribute to a fundamental understanding of the physical processes that determine the structure and properties of these materials. In the case of point defects, it is attractive to use quantum mechanics to describe the region of the crystal in proximity to the defect, perhaps embedding this region in an external potential determined by some auxiliary principle. The hope here is that the response of the lattice to the point defect may then be described by some method which is simpler than the quantum mechanical method used to describe the point defect itself. It is noted parenthetically, that the definition of simpler, as used here, is that it be computationally less expensive to use. Similar considerations apply in a similar way to the case of adsorbates on solid surfaces. Models which may accurately describe the response of an embedding lattice do currently exist. In the present case we begin our development for the case of non-metals. In many studies performed prior to the present for such systems, the use of a classical shell model, based upon point charges, and masses, interacting by simple parameterized potentials has been successful in correlating perfect-lattice equilibrium data with the ground state properties of defects in these systems.[1,2] Therefore, we begin our study by choosing to think of the embedding lattice in terms of the classical shell model. We find that it is possible to retain the functional form of the shell model, but determine all needed parameters from the quantum mechanical calculation, and to augment this functional form with appropriate angular potentials as well. The region about the defect can be described by means of an Unrestricted Hartree-Fock method.[3] (UHF) Such a model will in practice not yield sufficient accuracy for our purposes, and is extended by the implementation of a Many-Body Perturbation-Theory method (MBPT). By this choice, we will separate the problems of exchange from those of correlation, rather than combine them as is often the case in a computation based on the density functional method.[4]

In the case of a cluster embedded in a classical lattice, special care needs to be taken to ensure that mathematical consistency is achieved between the cluster and the embedding lattice. This has been solved formally by the work of Kunz and Klein,[5] who achieve this through the introduction of a localizing potential, here called the Kunz-Klein localizing potential or KKLP.

Simulation of a large crystallite or an infinite lattice containing a point defect represented by a cluster and a polarizable embedding lattice is implemented here by

means of an energy minimization procedure. That is, one minimizes the total system energy with respect to all parameters that define the lattice and the electronic configuration. For those parts of the lattice described by the shell model, one must minimize the total energy with respect to the positions of the ion cores, and also with respect to the polarization of the ions individually. For the quantum mechanical cluster, energy minimization is carried out with respect to the nuclear positions and also the electronic configuration. In this method it is possible to study states other than the ground state. Since the primary physical outputs are total energies and geometries, spectroscopic data is obtained from total energy differences. Positional variations are carried out initially using the HADES approach as implemented in the ICECAP procedure, or more recently using a Monte Carlo approach.

In the next section of this paper, we describe the basic theoretical ideas used in this study. This will include the shell model lattice, the UHF method, the KKLP and the inclusion of correlation via MBPT. We desired to find a simple molecular system which might illustrate the essential need of correlation in studies of binding behavior, and found such a case in the molecule KrF_2. In this case we find the molecule to be unbound in the UHF limit, but is strongly bound in the correlated limit. The strength of binding, is about 1 eV, and is suggestive that some form of an induced giant enhanced polarizability is responsible for this binding if one wishes to relate this binding to a van der Waals model. This induced polarizability must be generated when the atoms are in proximity to each other. The entire methodology is extended to considerations of the case of a simple point defect case. This is the substitutional impurity of a 3d transition metal in MgO. Ground state studies are reported here for V^{++}, Cr^{3+} and Mn^{4+}, with excited state results obtained as well for the case of Cr^{3+}. The more complex case of Cr^{3+} in Al_2O_3 is considered as well. In this case the Cr substitutes for an Al ion, and the lattice is found to relax asymmetrically in response to the presence of the Cr^{3+} impurity. Given the technological importance of optical properties of Ruby, this ability to predict asymmetric relaxations is quite significant.

2. Theoretical Methods.

In these studies, we assume that we have a system consisting of n electrons and N nuclei. The n electrons have coordinates designated by x_i and mass, m, and charge e. The nuclei have coordinate R_I, and nuclear charge Z_I. In these studies, the Born-Oppenheimer approximation is used and thus the nuclear mass is treated as infinite. The electron coordinate includes spin degrees of freedom. In general lower case letters refer to electron attributes, while upper case letters refer to nuclear properties. In this study the atomic system of units is used. That is; Plank's constant, the electronic charge, and the electronic mass are set to unity. Thus, the unit of length is approximately 0.53 x 10^{-8} cm, and the unit of energy is approximately 27.2 eV. In the usual non relativistic formalism, the Hamiltonian for the system is:

$$\mathcal{H} = -\frac{\hbar^2}{2m}\sum_{i=1}^{n}\nabla_i^2 - \sum_{i=1}^{n}\sum_{I=1}^{N}\frac{Z_I e^2}{|\vec{r}_i - \vec{R}_I|} + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n'}\frac{e^2}{|\vec{x}_i - \vec{x}_j|} + V_{NN} \qquad (1)$$

Ideally one would like to solve the n-electron Schroedinger equation for this Hamiltonian:

$$\mathcal{H}\Psi(\vec{x}_i, \vec{x}_j) = E\Psi(\vec{x}_i, \vec{x}_j) \qquad (2)$$

but computational difficulties preclude this. Instead we will resort to a series of approximations beginning with the UHF approximation. In the UHF approximation, the n-electron wavefunction is approximated by an antisymmetrized product of one electron orbitals. These orbitals are chosen to be orthonormal, and to minimize the energy expectation value of the Hamiltonian with respect to the functional form of these orbitals. This set of approximations leads to the system of equations called the UHF equations:

$$F(\rho)\,\varphi_i(\vec{x}) = \epsilon_i\,\varphi_i(\vec{x}) \qquad (3)$$

$$F = -\frac{\hbar^2}{2m}\nabla^2 - \sum_{I=1}^{N}\frac{Z_I e^2}{|\vec{r} - \vec{R}_I|} + e^2\int\frac{\rho(\vec{r}\,',\vec{r}\,')}{|\vec{r} - \vec{r}\,'|}d\vec{r}\,'$$

$$- e^2\rho(\vec{x},\vec{x}\,')\frac{1}{|\vec{r} - \vec{r}\,'|}\hat{P}(\vec{x},\vec{x}\,') \qquad (4)$$

$$\rho(\vec{x},\vec{x}') = \sum_{i < i_{fermi}} \varphi_i(\vec{x})\,\varphi_i^{\dagger}(\vec{x}') \tag{5}$$

$$\int \varphi_i^{*}(\vec{x})\,\varphi_j(\vec{x})\,d\vec{x} = \delta_{ij} \tag{6}$$

This series of equations may be solved in matrix form using a basis set expansion in terms of contracted gaussian basis orbitals. This procedure is so standard as to require no further discussion here.

The practical problem is to be able to solve this set of equations for extended systems. In the case of the pure, perfect, periodic system, techniques of energy band theory may be used[7]. However, we wish to be able to consider defects as well. There are methods to study some defect cases based upon periodic super cell methods[8], but in our case the study of charged defects in insulating solids is envisioned. Such studies don't lend themselves well to super cell methods due to the infinite range of the coulomb potential. Instead, we resort to the older method of local orbitals introduced formally by Adams[9], and Gilbert[10], and given a computational formulation by the author[11]. In this method, we formally divide the system into two parts, the cluster, and its environment. The cluster in practice contains the defect or impurity in question as well as the first few shells of atoms surrounding the defect. The environment contains the remainder of the system.

The Hamiltonian is formally partitioned into two parts, F_A, the cluster Hamiltonian, and V_A, the Hamiltonian for the environment. These are formally:

$$F_A = -\frac{\hbar^2}{2m}\nabla^2 - \sum_{I\,in\,A} \frac{Z_I e^2}{|\vec{r}-\vec{R}_I|} + \int \frac{\rho_A(\vec{r}',\vec{r}')}{|\vec{r}-\vec{r}'|}\,d\vec{r}' - e^2 \rho_A(\vec{x}',\vec{x}) \frac{1}{|\vec{r}-\vec{r}'|} \hat{P}(\vec{x}',\vec{x}) \tag{7}$$

$$V_A \doteq F - F_A \tag{8}$$

We may further divide the term V_A into two parts. The first is due to the ionic nature of the individual atoms, if any,

and is called V_M, and the part due to the non-ionic nature of the atoms, a short range part termed U_A. Having done this we may formally consider the Adams-Gilbert modified UHF equation:

$$[F + \rho A \rho]\hat{\varphi}_{Ai} = \hat{\epsilon}_{Ai}\hat{\varphi}_{Ai} \tag{9}$$

This equation is simply a canonical transformation on the original UHF equation[10], and its solution forms a first order density matrix identical to that of the original UHF equation, and leaves the total system energy unmodified. Thus:

$$\begin{aligned}\rho(\vec{x},\vec{x}') &= \sum_{i=1}^{n} \varphi_i(\vec{x})\varphi_i^{+}(\vec{x}') \\ &\equiv \sum_{iA}\sum_{jB} \hat{\varphi}_{Ai}(\vec{x})S^{-1}_{Ai,Bj}\,\hat{\varphi}^{+}_{Bj}(\vec{x}')\end{aligned} \tag{10}$$

and

$$S_{Ai,Bj} = \langle \hat{\varphi}_{Ai} | \hat{\varphi}_{Bj} \rangle \tag{11}$$

The presence of the overlap term is formally required since it is not required that each partitioning of the system have the same arbitrary function A used for it. Therefore, the various orbitals need not see the same Hamiltonian, and are therefore not necessarily orthogonal.

In the present implementation, one will chose the arbitrary operator, A, to be $-U_A$. Therefore, we solve the one-electron equation:

$$[F_A + V_M + U_A - \rho U_A \rho]\hat{\varphi}_{Ai} = \hat{\epsilon}_{Ai}\hat{\varphi}_{Ai} \tag{12}$$

In this implementation the term $\rho U_A \rho$ tends to cancel the term U_A in the original UHF operator, and allows the cluster to localize a number of electrons in it. The remainder of the electrons are of necessity delocalized into the environment. One may naturally play the same localization trick on the environment electrons as well.

The term $\rho U_A \rho$ is called the KKLP due to its operational effect. The strategy of solution is as follows: Begin with the pure, perfect, periodic lattice. Divide this

lattice into some set of natural building blocks. One's intuition will normally dictate a good choice of building block. For example, in the case of crystalline NaCl, a natural set of building blocks are the systems Na^+, and Cl^-. These are not free space ions but rather ions self consistently distorted by the crystalline environment. We assume that the orbitals associated with ions situated suitably far from the defect are the same as for the ions of the pure, perfect, periodic crystal. We do not assume that they sit at perfect lattice sites, and allow the potential due to them to be further modified by a polarization contribution. Using this environmental solution, the set of equations (12) may be solved for the cluster. Within the cluster, atomic positions and symmetries are relaxed to establish a minimum in total energy directly from the quantum mechanical problem. The atomic positions and polarizations are also relaxed in the environment to minimize total system energy. This is not done at the quantum level, however, due to the complexity of this system. Here we resort to use of the shell model. In this model, each core has a charge, as does each shell. If the atoms are ionized these charges are not equal. The shells interact with their core by a harmonic force, providing for atomic/ionic polarizability. The shells are set to interact with each other via a Buckingham potential in addition to the electrostatic potentials. The parameters of the Buckingham potential, the harmonic potentials and the core and shell charges may be found from fitting experimental data. Such is not a requirement, and these may be obtained directly by calculation. This then is a brief description of the essential features of the method termed ICECAP[12]. In the previously implemented ICECAP code the positional relaxation is accomplished by application of a conjugate gradient procedure. In more recent developments, we accomplish the positional relaxation by means of a Metripolous procedure within the Monte Carlo methodology. This offers several procedural advantages. In this method getting trapped in local minima rather than absolute minima is reduced as a hazard, the study of low symmetry situations is facilitated, and the inclusion of finite temperature effects is accomplished.

In the above discussion, we used the one electron approximation exclusively. This is found to be of inadequate precision for our needs and therefore, we seek to include explicit electron correlation in the cluster computation. This is most easily achieved by the use of a MBPT formalism. This is a natural choice in some ways for a solid system due to the simple fact that the MBPT method is extensive (size-consistent)[13].

The essential features of this approach are demonstrated by consideration of the non-degenerate state case. Extension to a degenerate system can be obtained as

well, although the practicalities of implementation are far less simple. In any event, for the cases needed here non-degenerate perturbation theory is adequate. Consider a Hamiltonian, H, which is partitioned into two parts, a zero order Hamiltonian, H_0, whose eigenvalues and eigenvectors are known, and a perturbation, V. Thus:

$$\mathcal{H} = \mathcal{H}_0 + V \tag{13}$$

and

$$\mathcal{H}_0 \Phi_i = w_i \Phi_i \tag{14}$$

From these solutions we may construct the eigenvalues, and eigenfunctions of the total Hamiltonian. That is we formally solve the correct equation:

$$\mathcal{H} \Psi_i = E_i \Psi_i \tag{15}$$

finding that

$$\Psi_i = \left[1 - (\mathcal{H}_0 - W_i)^{-1}(1 - P_I)(E - V - W_i)\right]^{-1} \Phi_i \tag{16}$$

and

$$E_I = W_I + \langle \Phi_I | V | \Phi_I \rangle + \langle \Phi_I | V (\mathcal{H}_0 - W_0)^{-1} (1 - P_I)(-V) | \Phi_I \rangle \tag{17}$$
$$+ \cdots$$

In the present case we need to properly pick a H_0. This is usually chosen in such calculations to be the sum of the one body UHF operator, as in Moeller-Plesset perturbation theory, however, we do not necessarily know the canonical UHF solution, only a canonical transformation of it. We therefore know only the eigenfunctions and eigenvalues of the Adams-Gilbert-Kunz equation (12). We chose the sum of these one body equations to be our zero order Hamiltonian. This allows a formally tractable solution to be obtained. This solution through second order becomes simply

$$E_I \approx \langle \Phi_i | \mathcal{H} | \Phi_I \rangle + \sum_{J \neq I} \frac{\langle I | V | J \rangle \langle J | V | J \rangle}{W_I - W_J} \tag{18}$$

This is the correlation correction used in our work In the next section, several examples of this method are discussed, including several simple cases to further graphically illustrate the utility of inclusion of correlation.

3. Some Test Cases.

A. KrF_2, a case of an Enhanced van der Waals Bond.

The inclusion of correlation effects in a solid state chemical calculation by explicit ab initio methods is both computationally expensive and difficult to formulate. Therefore it is desirable to give an illustration of the need for this refinement. In general the only obvious case is the bonding of molecular solids such as the solid rare gasses. In such a case the bonding per molecule/atom is so small that the bond doesn't survive the elevation of the system's temperature to room temperature. Such effects are masked by more dominant forms of bonding in most solid systems. We seek to find a case where correlation is dominant. Several such systems are thought to exist in the case of surface physics, for example the case of the chemisorption of rare gas atoms onto metal surfaces. In the case of the adsorption of Xe on W bonds of the size of 1.1 eV/atom exist. It is true that other systems exhibit adsorption strengths on the order of typical van der Waals bonds, which are several orders of magnitude less strong.[14,15,16] Here we seek a simpler molecular analog to demonstrate the need for correlation. Such a case is presented by KrF_2. The extension to the adsorption of rare gasses on metals is possible by like methods, and has been accomplished for several cases.[17]

The basis set chosen to represent Kr and F were obtained by using the best set from Huzinaga[18]. This set was immediately enhanced by splitting the outer s,p,d orbital on Kr into two basis functions, and by splitting the outer s, and p orbital on F into two basis functions. In addition s,p,d,f polarization primitive gaussians were added to the Kr set and polarization s,p,d primitives were placed on both fluorines, the exponents being chosen by nonlinear variation. This basis set was then used to study the possible geometry of the KrF_2 molecule. Initially the geometry was studied in a symmetric linear F-Kr-F system. In this geometry, it was determined that a minimum in total energy occurred when the Kr-F separation was 3.6 au. Having established this minimum, its stability was examined with respect to changing the F-Kr-F angle, and with respect to allowing the two Kr-F distances to vary independently. These studies indicated that the linear F-KR-F molecule with identical Kr-F separation was in fact a minimum on the potential energy surface.

The study was conducted for three initial wavefunction possibilities. The HF level trial function was permitted to be a triplet, a closed shell singlet (RHF), and an open shell singlet. For large Kr-F separation, the lowest energy state was the open shell singlet. This was a two determinant wavefunction, at the independent particle level. The UHF

single determinant wavefunction was triplet contaminated to the extent of 25%. The triplet lay a bit above the open shell singlet, and became degenerate with it by the time that the Kr-F separation became 10.0 au. For such large separations the closed shell singlet wavefunction lay substantially above the other two orbitals. However at short Kr-F distance the condition became otherwise. At a separation of about 3.9 au these states would have been essentially degenerate if such a degeneracy were allowed. For shorter distance the lowest energy state is the closed shell singlet. This state has a minimum energy at a separation of 3.4 au in the Hartree-Fock case and at 3.6 au in the correlated case. There is a substantial difference in the two cases however. The HF minimum at 3.4 au is unbound with respect to separation into atoms by 3.30 eV, whereas the correlated state is bound by an amount equal to 1.09 eV. Thus the inclusion of correlation effects enhances the energy of binding of KrF_2 by 4.49 eV, and changes the system into a bound molecule from an unbound one The potential energy curve for the KrF_2 molecule is seen with and without correlation for the symmetric linear case in table 1, for the three cases discussed.

It is possible to attempt to understand the large enhancement of the correlation energy of KrF_2. Near the bonding distance one finds that the lowest singlet has a second singlet state slightly above it in energy. This second singlet forms a strong configuration interaction with the lower one and promotes the enhanced correlation bond. Thus this is an example of a molecular system exhibiting a form of "giant enhanced van der Waals" bond, as well as a qualitative explanation for it.

TABLE 1. The lowest singlet state and the triplet state potential energy surface for a symmetric linear FKrF molecule is given. Energies are with respect to the isolated atom limit, and are in eV, while the Kr-F distance is in au.

R Kr-F	RHF Singlet		open shell Singlet		Triplet	
	HF	MBPT	HF	MBPT	HF	MBPT
3.2	3.83	-.20			na	na
3.4	3.39	-.96			7.14	4.98
3.6	3.60	-1.09			4.93	3.18
3.7	3.85	-1.01			4.09	2.57
3.8	4.17	-1.00			3.40	1.95
4.0			2.11	0.27	2.36	1.43
5.0			0.43	0.02	0.41	0.01
10.0			0.0	0.0	0.0	0.0

B. Monte Carlo Study of NaF.

Solid NaF was studied previously as a host system for the impurity Cu^+.[4] This study also deduced properties such as the lattice constant of NaF using the HADES approach. Buckingham potentials were used in this study. We repeat this study for this case using clusters of finite size, the same potentials and a Monte Carlo simulation. This is done as a function of cluster size. The individual atoms were allowed to move using a Monte Carlo method. The atom to be moved was chosen at random, as was the direction and length of the proposed motion. The decision to accept the proposed motion was then made using the Monte Carlo method. The only deviation from this ideal is that a maximum possible displacement is imposed. The equilibrium geometry as a function of ion numbers is seen in table 2, along with the relative computer run time. The smallest system is defined as unit run time. We note for completeness that due to the inclusion of unshielded ionic potentials, all ions interact with all others here and our algorithm can't benefit from use of finite range potentials.

TABLE 2. The equilibrium bond length determined by averaging all nearest neighbor bonds for finite clusters of NaF are given. Lengths are in au. The number of ions studied are included, and relative run times are given.

Total no. of ions used	Average 1st neighbor distance	Relative run time
8	4.01	1.0
27	4.31	1.1
64	4.30	1.4
125	4.28	6.5
216	4.31	17.0
512	4.34	95.5
1000	4.37	379.
2744	4.43	2331
experiment	4.40	

C. Cr^{3+} in MgO.

Luminescent transitions within the unfilled 3d shell of transition metal ions in insulating crystals are of much current interest for example as tunable four-level laser systems. In this section we chose to study the electronic properties of $3d^3$ impurities in MgO. One such impurity is the Cr^{3+} ion. The $3d^3$ impurities prefer to occupy octahedral sites in insulators as is known from EPR studies. The one-electron energy levels are described in terms of a doubly degenerate set of orbitals, the e_g orbitals, and a three fold degenerate set, the t_{2g} orbitals. The splitting of these orbitals is termed 10Dq in energy. In terms of the

many body system, the ground state of the d^3 system is an $^4A_{2g}$ state. The lowest lying excited states may arise out of a d-d transition and be $^4T_{2g}$ in character, or may arise out of a d-s transition and be $^4T_{1g}$ in character.

The molecular cluster embedded in the classical lattice consists of the transition metal ion and the 6 surrounding O^{--} ions at sites (a00). The gaussian basis sets for the 3d ion are obtained from Huzinaga.[18] In addition diffuse s,p,d basis functions were added to the 3d ion's set to allow the description of the excited state. The O^{--} sets are obtained using the procedure of Pandey and Vail.[19] The nearest-neighbor distance and the short-range potentials for the embedding lattice used in our ICECAP procedure are taken from Sangster and Stoneham.[20] The calculations performed are with the inclusion of MBPT corrections.

Initially one considers the isolated d^3 impurity embedded in MgO in its ground state. If the impurity is V^{++} then the lattice exhibits an outward displacement of 2% for the nearest neighbors in equilibrium. This relaxation adds a stability of 0.07 eV to the system. The case of Cr^{3+} is different. This is a positively charged center in the lattice. In this case the nearest neighbor O-- relax inward by 5% of the nearest neighbor distance, and the relaxation contributes 0.84 eV to the stabilization energy. The impurity Mg^{4+} causes an inward relaxation of 15% and further stabilizes the lattice by some 7.0 eV.

Excited state energies are calculated for the Cr^{3+} impurity in MgO. The Frank-Condon principle is used in these calculations. We find that the transition energies from the $^4A_{2g}$ ground state to the $^4T_{1g}$ and the $^4T_{2g}$ states respectively lie at 1.62 eV (490 nm) and at 2.50 eV (77 nm). Experimentally Okada et al. [21] have assigned a peak at 445 nm to the $^4A_{2g}$-$^4T_{1g}$ transition. A further peak found at 620 nm is ambiguous in its identification. Further studies are needed to identify the origin of this peak and to further resolve this spectrum.

D. The Cr^{3+} Impurity in Al_2O_3.

The Cr^{3+} ion substitutes for Al in the ruby lattice (Al_2O_3), in a way that is neutral with respect to the lattice unlike Cr in MgO. The ruby crystal has the O^{--} forming a hexagonal close packed sub-lattice. The Al ions on the other hand only occupy 2/3 of the available sites on the potentially close packed cation sub-lattice. Although the Cr ion is of like charge to the Al ion it replaces, the Cr^{3+} ion is substantially larger in ionic radius than is Al^{3+}. The radii are 0.064 nm and 0.050 nm respectively. Thus one may expect the Cr ion to dilate the lattice in its vicinity. This is found to be the case here. The lattice arraignment is best seen as a plane of O in a hexagonal close packed array, a plane of Al ions, a plane of O ions, a plane of Al ions one will be replaced by a Cr ion, a plane of O ions, a plane in which the Al ion is missing, and has an empty site,

and this structure repeated. Substitute a Cr ion for the Al ion indicated above and a strong asymmetric relaxation occurs. In this case the 3 O ions in the plane between the Cr ion and the Al ion relaxes 0.6% of a lattice plane separation toward the Al ion plane, whereas the 3 O ions in the plane between the Cr ion and the empty site relax by an amount of 3.7% toward the empty site. This result is significant in that it verifies the result deduced from experiment in which ruby crystals with high Cr concentrations exhibit asymmetric relaxations about the Cr site, namely that this relaxation would also occur for the case of dilute Cr concentration as well. The present case of only a single Cr impurity represents the limit of the dilute impurity case

4. CONCLUSIONS.

The technique which we term ICECAP for studying the properties of defects in solids has been seen to produce useful quantitative predictions of spectroscopic as well as ground state data. We have also seen that the model is capable of modification to use a Monte Carlo approach for the determination of geometric positions of atoms as well as their polarization. This extension allows a simple extension of the ICECAP technology to the case of low symmetry or to systems with line or planar defects (eg surfaces, interfaces, grain boundaries). The model is systematically capable of development so that all parameters needed for the HADES model portion of the ICECAP procedure are capable of being obtained from theoretical calculations rather than from experiment as has been done initially.[17] It is further seen that so doing has the advantage of not only extending this procedure to systems for which a data base is nonexistent but also extending the procedure to cases for which potential parameters obtained from experiment are not well adapted to distortions far from equilibrium. Thus, the theoretical determination of atomistic modelling parameters is more accurate for the determination of the an-harmonic part of the potential than the use of experimental data. The inclusion of correlation corrections are seen as essential if one is to achieve high precision numerical estimates of spectroscopic and ground state properties. We see that correlation bonding far stronger than usual van der Waals bonds is possible and likely due in strength due to near degeneracy of several low-lying singlet configurations. We also find that studies of systems with complex ground states such as the Cr^{++} impurity are easily accomplished, and that asymmetric lattice relaxations in response to impurities may be described within this theory.

This research is supported by the U. S. Navy Office of Naval Research under grant ONR N-00014-89-J-160.

REFERENCES

1. B.G. Dick and A.W. Overhauser, Phys. Rev. 112, 90 (1958).
2. H. Bilz and W. Kress, Phonon Dispersion Relations in Insulators (Springer-Verlag, Berlin, 1979).
3. A.B. Kunz and J.M. Vail, Phys. Rev.B38, 1058 (1988).
4. A. B. Kunz, J. Meng and J.M. Vail, Phys. Rev. B38 1064 (1988).
5. A.B. Kunz and D.L. Klein, Phys. Rev. B17, 4614 (1978).
6. J.H. Harding, A.H. Harker, P.B. Keegstra, R. Pandey, J.M. Vail and C. Woodward, Physica B+C131B, 151, (1985).
7. J.C. Callaway, Quantum Theory of the Solid State, (Academic Press, New York, 1974).
8. A.M. Stoneham, Theory of Defects in Solids, (Oxford Press, Oxford, 1975).
9. W.H. Adams, J. Chem. Phys., 34, 89(1961).
10. T.L. Gilbert, in: Molecular Orbitals in Chemistry Physics and Biology, Ed. P.O. Lowdin,(Academic Press, New York, 1964).
11. A. B. Kunz, phys. stat. solidi, 36, 301, (1969).
12. R. Pandey and A.B. Kunz, J. Mater. Res., 3, 1362, (1988).
13. D.J. Thouless, Quantum Mechanics of Many Body Systems, (Academic Press, New York, 1961).
14. J.C.P. Mignolet, J. Chem. Phys., 21, 1298, (1953).
15. G. Ehrlich and F.G. Hudda, J. Chem. Phys., 30, 493, (1959).
16. T. Engel and R. Gomer, J. Chem. Phys., 52, 5572, (1970).
17. A. B. Kunz, Int. J. Quant Chem. Symp. 24, 24, 607, (1991).
18. S. Huzinaga, Gaussian Basis Sets for Molecular Calculations (Elsevier, New York, 1984).
19. R. Pandey and J. M. Vail, J. Phys. C 1, 2801, (1989).
20.M. J. Sangster and A. M. Stoneham, Phil. Mag. B43, 597,(1981).
21. Okada et al., Cryst. Latt. Def. and Amorph. Mat., 17, 57, (1987).

STUDENTS SUPPORTED BY THIS GRANT

1. Philip Kaldon, PhD 1989.
2. Sha Wang, MS 1990.
3. Jun Zuo, PhD 1991.
4. Guan Gao, PhD expected 1991.

HONORS

1. A. B. Kunz, Editorial Board, International Journal of Quantum Chemistry

2. A. B. Kunz, Editorial Board, Computers in Physics

3. A. B. Kunz, Scientist of the Year Award, Impressions 5 Science Museum

PUBLICATIONS SUPPORTED BY THIS GRANT

Copies of the first page of all articles supported by this grant are included after this page.

Derivation of interionic potentials using embedded quantum-mechanical clusters: Cation and anion impurities in MgO

Ravindra Pandey, Jun Zuo, and A. Barry Kunz
Department of Physics, Michigan Technological University, Houghton, Michigan 49931

(Received 6 September 1989; accepted 13 November 1989)

The ICECAP methodology is used to derive interionic potentials of some cation and anion impurities in MgO, namely, Li^+, Na^+, K^+, Be^{2+}, H^-, S^{2-}, and O^{2-}. Analysis is given of the defect energies obtained by using the derived impurity potentials. Based on the available experimental data, comparison is made to justify the reliability of the derived impurity potential for Be^{2+}. The calculated activation energy for Be^{2+} diffusion comes out to be 1.54 eV as compared to the experimental value of 1.60 eV, which is considered to be very satisfactory.

I. INTRODUCTION

There is significant interest in developing reliable interionic potentials for ionic systems, most importantly the oxides and related ceramic materials. Magnesium oxide (MgO) is such a technologically important ceramic with applications ranging from catalysis to microelectronics. It is a simple oxide of the NaCl structure and has therefore been considered as the prototypical oxide for both experimental and theoretical studies of defect properties of ceramic materials.

Interionic potentials of ionic crystals are generally derived from empirical fittings to perfect lattice properties, such as cohesive energies, elastic and dielectric constants, ensuring that the potentials are compatible with lattice stability.[1] This approach, however, does not provide impurity ionic potentials directly since it relies heavily on the availability of experimental data. One therefore uses an arbitrary averaging method to extract impurity potentials from host lattice potentials. An alternative, nonempirical approach is to obtain the potentials by using electron gas methods. Here the interaction between charge densities representing the interacting ions is calculated, the densities being obtained by calculating the wave function of the isolated ion.[1] This method, however, approximates the exchange and correlation potentials and does not allow the distortion of the charge densities which is expected for the cases of highly polarizable anions. Thus, the derivation of reliable impurity potentials has so far proved to be a difficult task.

With the availability of the ICECAP program package,[2] we have undertaken a study to derive reliable impurity interionic potentials in ionic crystals. In earlier works, we derived the potentials for the impurities, namely Cu^+ and Ag^+ in some alkali halides, concluding that a more accurate derivation would require a larger distortion of the embedded cluster.[3,4] In the present work, we derive interionic potentials for impurities using large distortions (typically about 25%) in the cluster and then use them to obtain defect energies in MgO. The impurities considered here are Li^+, Na^+, K^+, Be^{2+}, H^-, and S^{2-}, substituting the host cation/anion in MgO.

Our approach has been to dilate and compress the quantum-mechanical cluster containing the impurity. Interionic potentials are then determined such that the same sequence of distortions, applied to a shell-model lattice containing the impurity, produces the same energy variation. In all cases, the embedding lattice is fully relaxed. In Sec. II we will give a brief description of this method of deriving impurity potentials. In Sec. III we will present and discuss our results, including the derived potentials, defect energies, and impurity diffusion in MgO.

II. METHOD

We simulate impurity-doped MgO in ICECAP calculations as a molecular cluster consisting of the substitutional impurity, its nearest-neighbors, and/or second-nearest neighbors embedded in the lattice represented by the shell model.[5] ICECAP combines electronic structure calculations with shell-model treatment of lattice polarization and distortion, with the electronic structure and lattice relaxation components being integrated self-consistently. The ICECAP methodology is described in detail in a variety of other papers.[2,6,7] The unrestricted Hartree-Fock self-consistent field (UHF-SCF) approximation is employed to describe the electronic structure of the molecular cluster.[8]

In the shell model, each point-ion consists of a core of charge X and a shell of charge Y, such that the total ionic charge is the sum of the core and shell charges. The ionic polarization is described by the displacement of a massless shell from a massive core, the two being connected by a harmonic spring with a force constant K. The polarizability of an ion is then given by Y^2/K. The interionic potential energy may then be expressed as a sum of pairwise terms of the form:

19

PHYSICAL REVIEW B VOLUME 41, NUMBER 11 15 APRIL 1990-I

Ab initio study of localization and excitation of an excess electron in alkali halide clusters

Ravindra Pandey, Max Seel, and A. Barry Kunz
Physics Department, Michigan Technological University, Houghton, Michigan 49931
(Received 5 February 1990)

Hartree-Fock calculations coupled with second-order many-body perturbation theory have been performed to study binding energies, localization, and excitation properties of an excess electron in various alkali halide clusters, Na_nF_{n-1}, Na_nCl_{n-1}, and Li_nF_{n-1} ($n=2, 4, 5, 14$). The binding energies agree well with recent experimental data and three different modes of localization are corroborated. The position of the F-center absorption band in Na_nF_{n-1} clusters is verified, but not for Na_nCl_{n-1}. New absorption bands for Na_nCl_{n-1} and Li_nF_{n-1} clusters are predicted.

The interest in the physics and chemistry of small clusters is rapidly increasing due to their novel and hybrid properties. Recent experimental[1-3] and theoretical[4,5] work focused on the properties of an excess electron attached to a cluster since the extra electron influences the cluster stability and therefore the reactive properties. Honea *et al.*[1] have measured abundances and binding energies of an excess electron interacting with $(Na_nF_{n-1})^+$ clusters. Based on observed abundances and ionization threshold they classified the clusters as follows: cubic clusters consist of a filled cubic lattice of ions with the extra electron occupying a weakly bound surface state; F-center clusters consist of a nearly filled cubic lattice with an electron localized in an anion vacancy; and noncubic clusters have the excess electron bound to a single cation. The F-center and the noncubic clusters show high electron binding energies. As further evidence for electron localization, Honea *et al.* cite the observation of strong optical-absorption bands in F-center clusters using resonant two-photon ionization spectroscopy.

In this paper we report results of an *ab initio* study of localization and excitation properties of an excess electron attached to various alkali halide clusters and provide a basis for the relation between binding energies, excitation energies, and the degree of localization. With respect to localization properties, we find good agreement with earlier theoretical predictions[4,5] based on quantum path-integral molecular-dynamics calculations and corroborate the interpretation given in Ref. 1. The calculated binding energies agree well with the experimental data, and, for the noncubic cluster Na_5F_4, are in better agreement than the binding energies obtained from a cruder model.[1] For the excitation energy we verify the position of the F-center absorption band in Na_nF_{n-1} clusters, but cannot support the interpretation that the observed blue-green band in Na_2Cl cluster is associated with the excitation of the excess electron.

Calculations are performed for various Na_nF_{n-1}, Na_nCl_{n-1}, and Li_nF_{n-1} clusters where $n=2, 4, 5$, and 14. The internuclear separation between cation and anion in the cluster is taken to be the same as in bulk solid which is 3.80, 4.36, and 5.31 bohrs for LiF, NaF, and NaCl clusters, respectively. [Preliminary cluster-geometry optimization—only bond lengths, but not bond angles—shows the lowering of total energy only in the case of "ionized clusters." For example, the $(Na_2Cl)^+$ cluster relaxes inward to the bond length of 4.81 bohrs, relative to the bulk value of 5.31 bohrs, lowering the total energy by 0.1 eV. No relaxation of cluster geometry from the bulk separation has been found for the neutral clusters.]

The unrestricted Hartree-Fock linear combination of atomic-orbitals method is employed. Correlation corrections are calculated using second-order many-body perturbation theory.[6] For the expansion of the atomic orbitals for Na, F, and Cl, Huzinaga Gaussian basis sets[7] are split into contractions of (421/4), (421/4), and (4321/43), respectively. For Li, a (6,1) basis set[8] is used. The excess electron in the F-center cluster is accommodated by adding another single Gaussian whose exponent is determined variationally.

Table I gives the binding energy of the excess electron in various alkali halide clusters. The binding energy is defined as the difference between total cluster energies of a neutral cluster, for example, Na_nF_{n-1}, and the ionized cluster, for example $(Na_nF_{n-1})^+$. The total cluster energies are the values obtained from Hartree-Fock calculations coupled with second-order many-body perturbation theory. As it turned out, the correlation corrections are

TABLE I. Binding energy of the excess electron in alkali halide clusters calculated by Hartree-Fock coupled with second-order many-body perturbation theory.

	Binding energy (eV)		
	This work	Observed[a]	Calculated[a]
F-center clusters			
Na_2Cl	4.06	...	...
Na_4Cl_3	4.17	...	...
Na_2F	4.29	3.85+0.15	3.50
Na_4F_3	4.28	3.54+0.15	3.80
Li_2F	4.60	...	...
Li_4F_3	4.73	...	...
Noncubic cluster			
Na_5F_4	3.80	3.85+0.15	3.10
Li_5F_4	4.33	...	...
Cubic cluster			
$Na_{14}F_{13}$	...	1.88	...
$Li_{14}F_{13}$	1.20	...	...

[a]Reference 1.

147 (3pp) 9c

J. Phys. Chem. Solids Vol. 51, No. 8, pp. 929–931, 1990
Printed in Great Britain.

0022-3697/90 $3.00 + 0.00

CHARACTERIZATION OF FLUORINE-DOPED MAGNESIUM OXIDE: A COMPUTER SIMULATION STUDY

RAVINDRA PANDEY and A. BARRY KUNZ
Department of Physics, Michigan Technological University, Houghton, MI 49931, U.S.A.

(*Received* 30 *October* 1989; *accepted* 24 *January* 1990)

Abstract—A computer simulation study is performed to characterize F^--doped MgO. The impurity potentials, namely F^-–Mg^{2+} and F^-–O^{2-} are derived using ICECAP and are then used to study F^- diffusion in MgO. The activation energy by vacancy mechanism comes out to be 1.53 eV. The excitonic state associated with the F^- ion is also studied. Furthermore, the excess electron associated with the F^- ion is predicted to be unbound in the lattice.

Keywords: Magnesium oxide, computer simulation, diffusion, charge states.

1. INTRODUCTION

Fluorine ions are found to be effective in promoting densification of MgO during hot-pressing or sintering [1]. Furthermore, an F^- ion substituting O^{2-} in MgO can be considered as an analog of the fission product, I^- in nuclear fuel oxides such as UO_2. Thus there is a significant interest in characterizing fluorine-doped MgO and in this paper we intend to achieve this objective by studying optical and transport properties of fluorine ion along with its different charge states in MgO.

Our simulation procedure is based on the embedded cluster model using the program package ICECAP (ionic crystals with electronic cluster, automatic program) [2]. The ICECAP procedure with its long-range lattice relaxation capability is ideal for such a study. In Section 2 we give a brief description of our computational model simulating the impurity, F^- in MgO. The results are presented and discussed in Section 3, and summarized in Section 4.

2. COMPUTATIONAL MODEL

The impurity, fluorine, in MgO is considered to occupy the on-center position as F^- substituting in the lattice for the O^{2-} ion. Since MgO has rock-salt structure with octahedral site symmetry, F^--doped MgO is modeled in ICECAP as a defect cluster of F^- ions at the cluster center, six nearest-neighbor Mg^{2+} ions at the sites (a, o, o) and/or 12 next-nearest-neighbor O^{2-} ions at the sites (a, o, o) where a is the nearest-neighbor spacing. In this way, the electronic structure of the F^- ion and all the neighboring ions that are assumed to be significantly affected by the F^- ion are described quantum-mechanically.

The defect cluster is embedded in the classical shell model lattice [3]. The cluster is therefore seen by its environment as a Coulomb potential and also by means of short-range interactions of the cluster ions with the shell model ions. The harmonic distortion and polarization of the embedding lattice are then determined by simulating the defect cluster by a set of point charges whose low-order electrostatic multipole moments match those of the defect cluster. ICECAP therefore combines electronic-structure calculations with the shell model treatment of lattice polarization and distortion in a mathematically and physically consistent way. (For a detailed discussion, we refer to Harding *et al.* [2] and Pandey and Vail [4].)

In the shell model, each point ion consists of a core of charge x and a shell of charge y, such that the total ionic charge is the sum of core and shell charges. The ionic polarization is described by the displacement of a shell from a massive core; the two being connected by a harmonic spring with a force constant k.

ICECAP applies the minimum energy principle to an infinite crystal of MgO containing the defect. The total defect crystal energy is minimized with respect to all shell and core positions and simultaneously with respect to variational parameters in the defect cluster wave function. This minimization is updated while the nuclear positions of the defect cluster are varied to give overall minimization of the total defect crystal energy. That is,

$$\frac{\partial E}{\partial R} = \frac{\partial E}{\partial \sigma} = \frac{\partial E}{\partial R_c} = 0. \tag{1}$$

This results in obtaining lattice (R), electronic (σ), and cluster (R_c) configurations and the total defect crystal energy E.

To describe the electronic structure of the defect cluster, we use the unrestricted Hartree–Fock self-consistent field (UHF-SCF) approximation obtaining

19c

PHYSICAL REVIEW B VOLUME 42, NUMBER 5 15 AUGUST 1990-I

Partial densities of states for silver bromide and silver iodobromide

M. G. Mason and Y. T. Tan
Research Laboratories, Eastman Kodak Company, Rochester, New York 14650

T. J. Miller and G. N. Kwawer*
Department of Physics, University of Illinois at Urbana-Champaign, 1110 West Green Street, Urbana, Illinois 61801

F. C. Brown
Department of Physics (FM-15), University of Washington, Seattle, Washington 98195

A. B. Kunz
Department of Physics, Michigan Technological University, Houghton, Michigan 49931
(Received 12 March 1990)

The valence-band photoemission of silver bromide and silver iodobromide has been measured with use of synchrotron radiation in the region of the Ag $4d$ Cooper minimum. The large change in ionization cross section in this region permits the determination of the individual halogen p and silver $4d$ partial densities of states (PDOS). The energy-distribution curves (EDC's) were recorded at liquid-nitrogen temperatures to prevent photolysis and take advantage of the significant line narrowing which occurs in the silver halides at low temperatures. The results are in good agreement with experiments using rare-gas resonance lines and with previously calculated energy-band structures. Changes in the halogen PDOS with the addition of iodide indicate that the narrowing of the band gap is due to the broadening of the uppermost antibonding halogen bands. The PDOS were calculated for pure AgBr using a nonrelativistic, self-consistent, Hartree-Fock theory and show good agreement with the experimental results.

I. INTRODUCTION

In the alkali halides the most loosely bound electrons on the halogen and the alkali-metal ions are well separated in energy.[1-3] This leads to a relatively simple valence band composed almost exclusively of halogen p-like orbitals. In the noble-metal halides the situation is significantly complicated by the presence of the metal d orbitals which are in near degeneracy with those of the halogen.[4-18] This near degeneracy leads to strong hybridization and considerable complexity in the valence-band structure. For example, in the rock-salt-structure materials, AgCl and AgBr, this orbital mixing has profound effects, causing these materials to have indirect band gaps and large valence-band widths. In the context of photoemission spectra, this mixing means that valence-band energy-distribution curves (EDC's) will be a composite of both the halogen and metal partial densities of states (PDOS's). The relative proportions will vary according to the energy-dependent ionization cross sections. In an ideal case, for each type of orbital composing the valence band there would exist a photon energy or energy range where its contribution to the experimental spectrum would dominate. By recording spectra at each of these energies, all of the constituent partial densities of states could be directly measured. This ideal situation is closely approximated in the Cu-Au alloys.[19] At low photon energies the EDC's reflect mainly the Au $5d$ density of states, but near the Au $5d$ Cooper minimum, at about 160 eV, the EDC's are due primarily to the Cu $4d$ density of states. As pointed out by Wertheim,[19] this procedure should have wide applicability for compounds or alloys containing $4d$-or $5d$-group elements.

In the silver halides the procedure for determining partial densities of states is somewhat more complex than in the copper-gold alloys. Whereas the valence Ag $4d$ orbitals do show a significant Copper minimum at around 130 eV,[20-23] the halogen ionization cross sections are not large enough to produce a spectrum characteristic of the pure halogen density of states.[24] It is, however, still possible to extract the partial densities of states from measured EDC's. It is only necessary that the ionization cross sections be known and that the relative values change significantly with photon energy. This procedure was pioneered by Cardona and co-workers and has been applied to the silver[9,10] and cuprous halides,[9] as well as the ternary compounds $AgInTe_2$ and $CuInS_2$.[25] More recently, the partial density of states in $Cu_{75}Pd_{25}$ has been determined with this method from synchrotron-radiation studies.[26] The basic assumption is that the experimental intensity, $N(E,h\nu)$, where E is the electron kinetic energy and $h\nu$ is the photon energy, is related to the ionization cross section per electron, $\sigma_i(h\nu)$, and the partial density of states, $\rho_i(E)$, by the simple relationship

$$N(E,h\nu)=C(E,h\nu)[\rho_p(E)\sigma_p(h\nu)+\rho_d(E)\sigma_d(h\nu)] . \quad (1)$$

The proportionality constant, $C(E,h\nu)$, contains experimental variables such as photon flux and analyzer transmission function as well as effects due to the electron

19 d

Cluster Modeling of Solid State Defects and Adsorbates: Beyond the Hartree–Fock Level

A. BARRY KUNZ

College of Engineering, Michigan Technological University, Houghton, Michigan 49931

Abstract

The use of finite clusters of atoms to represent the physically interesting portion of a condensed matter system has been an accepted technique for the past two decades. Physical systems have been studied in this way using both density functional and Hartree–Fock methodologies, as well as a variety of empirical or semiempirical techniques. In this article, the author concentrates on the Hartree–Fock based methods. The attempt here is to construct a theoretical basis for the inclusion of correlation corrections in such an approach, as well as a strategy by which the limits of a finite cluster may be transcended in such a study. The initial appeal will be to a modeling approach, but methods to convert the model to a self-contained theory will be described. It will be seen for the case of diffusion of large ions in solids that such an approach is quite useful. A further study of the case of adsorption of rare gas atoms on simple metals will demonstrate the value of inclusion of electron correlation.

Introduction

Lattice defects in or on crystalline materials, often in combinations that are difficult to resolve experimentally, determine many technologically important properties. Reliable computer simulations of such defects are of potential value, and may be expected to contribute to a fundamental understanding of the physical processes that determine the structure and properties of these materials. In the case of point defects, it is attractive to use quantum mechanics to describe the region of the crystal in proximity to the defect, perhaps embedding this region in an external potential determined by some auxiliary principle. The hope here is that the response of the lattice to the point defect may then be described by some method which is simpler than the quantum mechanical method used to describe the point defect itself. It is noted, parenthetically, that the definition of simpler, as used here, is that it be computationally less expensive to use. Similar considerations apply in a similar way to the case of adsorbates on solid surfaces. Models which may accurately describe the response of an embedding lattice do currently exist. The problem becomes to select one, and define fundamentally how to link such a model with the quantum mechanical cluster model. In the present case we begin our development for the case of nonmetals. In many studies performed prior to the present for such systems, the use of a classical shell model, based upon point charges and masses, interacting by simple parametrized potentials, has been successful in correlating perfect-lattice equilibrium data with the ground state properties of defects in these systems [1,2].

International Journal of Quantum Chemistry: Quantum Chemistry Symposium 24, 607–619 (1990)

CCC 0020-7608/90/010607-13$04.00

Ab initio band-structure calculations for alkaline-earth oxides and sulfides

Ravindra Pandey, J. E. Jaffe, and A. Barry Kunz
Department of Physics, Michigan Technological University, Houghton, Michigan 49931
(Received 28 September 1990)

The electronic structure of the oxides and sulfides of Mg, Ca, and Sr is computed with use of the self-consistent Hartree-Fock method including correlation. Energy-band structure and density of states are presented and discussed in context with the available experimental and theoretical studies. Our results predict that these materials (except MgS) are direct-band-gap materials.

I. INTRODUCTION

Alkaline-earth oxides are technologically important materials with applications ranging from catalysis to microelectronics. Alkaline-earth sulfides have been proposed as host materials for device applications such as multicolor thin-film electroluminescent and magneto-optical devices.[1]

Recently, Kaneko and his co-workers[2] have measured the optical spectrum of Ca, Sr, and Ba chalcogenides. They have interperted their results on the basis of a self-consistent augmented-plane-wave (APW) band-structure calculation concluding that these materials, except BaO, are indirect-band-gap materials with the lowest direct band gap at the X point. However, a more detailed look at the experimental results and their interpretation reveals some inconsistencies.

(i) The appearance of the two groups of peaks (assigned to the excitons at the X point and Γ point, respectively) in the [imaginary part of $\varepsilon(\omega)$] ε_2 spectrum showed no systematic trend in the oxides. It was absent in CaO and BaO, but was present in SrO.

(ii) It appears that the direct band gap in these materials was estimated from the ε_2 spectrum without taking account of the excitonic binding energy. [see Tables I and II of Ref. 2(a)].

(iii) None of the peaks in the ε_2 spectrum of CaO was assigned to the Γ_{15}-Γ_1 transition, but the assignment for the higher-order transition Γ_{15}-Γ_{25}' and Γ_{15}-Γ_{12} was given.

It is well known that band-structure calculations based on the local-density approximation (LDA) underestimate both the band gap and the valence-band width. Furthermore, the drastic lowering of the d-like conduction level (relative to the experiment) at the X point (i.e., X_3) has been observed in the LDA results for ionic materials such as NaCl.[3] Hence we believe that the reported interpretation of the optical spectrum has not properly taken account of the inherent limitations of the LDA-based calculations and is therefore somewhat ambiguous.

To provide a more accurate basis for the interpretation of the optical spectrum of these materials, we have undertaken a detailed and systematic investigation of the electronic structure of alkaline-earth chalcogenides using the Hartree-Fock method. This method has been highly successful in describing the electronic structure of alkali and silver halides.[3] The present work focuses on the nature of the energy gap of the oxides and sulfides of Mg, Ca, and Sr only. In the next section, we give a detailed account of the Hartree-Fock method including electron-correlation effects. In Sec. III the results are presented and compared to earlier studies involving both theory and experiment. Finally, conclusions are given in Sec. IV.

II. THEORETICAL METHOD

The basic method is Hartree-Fock and we begin with the canonical Fock equation,

$$F\phi_i(\mathbf{k},\mathbf{x}_i)=\varepsilon_i(\mathbf{k})\phi_i(\mathbf{k},\mathbf{x}_i)\ , \tag{2.1}$$

where the one-electron orbitals, ϕ's, are constrained to be orthonormal and eigenstates of the z component of spin and all pertinent crystal-symmetry operations. The Fock operator F is given by

$$F=\frac{-\hbar^2}{2m}\nabla^2-\sum_I\frac{Z_Ie^2}{|\mathbf{r}-\mathbf{R}_I|}+e^2\int\frac{\rho(\mathbf{x}',\mathbf{x}')}{|\mathbf{r}-\mathbf{r}'|}d\mathbf{x}'-e^2\int\frac{\rho(\mathbf{x},\mathbf{x}')}{|\mathbf{r}-\mathbf{r}'|}P(\mathbf{x}',\mathbf{x})d\mathbf{x}\ , \tag{2.2}$$

We note here that the Fock operator is a unique functional of the first-order density matrix ρ, which is given by

$$\rho(\mathbf{x},\mathbf{x}')=\sum_{\mathbf{k}}\sum_i\phi_i^*(\mathbf{k},\mathbf{x})\phi_i(\mathbf{k},\mathbf{x}')\ , \tag{2.3}$$

where the sum is carried over all occupied orbitals.

For the N-electron-system ground state, the occupied one-electron orbitals, ϕ's are the ones with the N lowest values of the Fock eigenvalue $\varepsilon_i(\mathbf{k})$. In the context of Koopemans's theorem, the eigenvalue of an occupied orbital $\varepsilon_i(\mathbf{k})$ is the negative of the energy needed to remove the electron (occupying the ith orbital) from the crystal, and the eigenvalue $\varepsilon_a(\mathbf{k})$ for a virtual (unoccupied) orbitals is the negative of the energy gained by adding an electron to the crystal. In both cases, the electronic density of the remaining electrons is unrelaxed. Hence the physics here refers to ionization properties, not to excitation properties of the N-electron system.

The self-consistent solution of the Fock equation (2.1)

195

A Study of Lithium Trapped-Hole Center in MgO Using Hartree-Fock Clusters

Jun Zuo, Ravindra Pandey, and A. Barry Kunz

Department of Physics

Michigan Technological University

Houghton, MI 49931

199

Correlated Hartree-Fock Electronic Structure of ZnO and ZnS

J. E. Jaffe, Ravindra Pandey and A. B. Kunz

Department of Physics, Michigan Technological University, Houghton, Michigan 49931

The band structures of ZnO and ZnS have been calculated by an all-electron Hartree-Fock method including correlation corrections. The goal is to evaluate the applicability to polar semiconductors of this highly efficient computational method, which was originally designed for closed-shell ionic systems, and to study the role of Zn 3*d*-band states in the electronic and optical properties of these materials. Comparison is made to the results of other calculations and to optical and photoemission data.

19h

Cu^+ in Alkali Halides:

Lattice Relaxation

Jun Zuo
College of Engineering
Michigan Technological University
Houghton, MI 49931

Ravindra Pandey
Department of Physics
Michigan Technological University
Houghton, MI 49931

Albert Barry Kunz
College of Engineering
Michigan Technological University
Houghton, MI 49931

Study of Excitonic States in Magnesium Oxide

Ravindra Pandey, Jun Zuo and A. Barry Kunz
Department of Physics
Michigan Technological University
Houghton, Michigan 49931

APPENDIX A
LISTING OF THE MONTE CARLO COMPUTER CODE

Jun Zuo, Author

Input file


```
THIS IS THE BEGINNING OF THIS FILE.
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

Monte Carlo simulation: NaF

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

THE PRESET PERCENTAGE OF ACCEPTANCE: 20
THE ACCURACY OF THE POSITIONS:       1.0e-2

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

THE NUMBER OF IONS PER SIDE: 10
PRINT OUT ALL COORDINATES? Y

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$??????????$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

THE INITIAL NEAREST NEIGHBOR SPACING (  Bohr    ): 4.5

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$????????$$$$$$$$$$$$

LATTICE ION              CHARGE             POLARIZABILITY (angstrom**3)
Na                         1                    0.196
F                         -1                    0.98

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

THE CENTRAL ION: Na

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

NUMBER OF DEFECT IONS: 0

DEFECT ION      CHARGE    POLARIZABILITY    POSITION RELATIVE TO THE CENTRAL ION

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

THE SHORT RANGE POTENTIAL: B*EXP(-R/RHO) - C/R**(6) + 1000000*CUT(D)

PARAMETERS AND UNITS:
B: eV         RHO: Angstrom   C: eV*(Angstrom)**6      D: Angstrom

                        B            RHO           C             D

Na <> Na             7895.4        0.1709        11.68         0.8277
Na <> F              1594.2        0.2555        0             0.1612
F  <> F              1127.7        0.2753        11.68         1.0208

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
THIS IS THE END OF THIS FILE.
```

Declaration and common block

```
      parameter(nmax=21)
      parameter(ndef=nmax*nmax*nmax)
c
      character*2 chn(100)
      character*3 norn(50)
      character*80 center_ion,title
      character*80 defect_site(2),ion(4)
      integer*4 ichoose(5,5),iorder(ndef),kind(ndef),move(ndef)
      real*4 a(15),alpha(10),b(10),c(10),charge(5),d(10),
     1p(3,ndef),potential(1000000,10),random1(97),random2(97),
     2rho(10),sroot(1000000),x(ndef),xtemp(ndef),y(ndef),
     3ytemp(ndef),z(ndef),ztemp(ndef)
      real*8 dum
c
      common /character/center_ion,title
      common /chaarray/chn,defect_site,ion,norn
      common /integer/icenter,ionedim,ipc,kind_of_center,l,maxtrial,
     1maxtrialp,move_total,move_totalp,myabove,myback,mybelow,myfront,
     2myleft,myright,n_defects,n_kinds,n_per_layer,n_per_side,
     3n_points,n_points_m1,n_points_m2,n_trials,n_trialsp,n100
      common /iran/ia1,ia2,ia3,ic1,ic2,ic3,iseed,ix1,ix2,ix3,
     1m1,m2,m3,negone
      common /iarray/ichoose,iorder,kind,move
      common /real1/accuracy,delta_e,all_charge,e_moveion,e_moveionp,
     1e_total
      common /real2/pre_poa,range
      common /real3/spacing,trials,xnew,ynew,znew
      common /rran/rm11,rm12,rm2,rm3,random1,random2
      common /array1/a,alpha,b,c,charge,d,rho
      common /array2/p
      common /array3/potential
      common /array4/sroot
      common /array5/x
      common /array6/xtemp
      common /array7/y
      common /array8/ytemp
      common /array9/z
      common /array10/ztemp
```

Main Program


```
      PROGRAM MC
c
c  The program picks one ion at a time randomly and tries a new position. All
c  movable ions are tried in one iteration. The old positions are replaced with
c  the new ones at the end of each iteration.
c
c  To keep the lattice from rotating, limitations are imposed on two ions so
c  that they can move only on a plane. The first one is the one in front of the
c  central ion, which can move only on the x-y plane. The second is the one
c  above the central ion, moving only on the x-z plane.
c
c  After each iteration (one trial for each movable ion), the order by which the
c  ions are moved is shuffled randomly.
c
c  When a new position is determined, it is stored in a temporary buffer. The
c  old position is kept until all movable ions are tried. This treatment is
c  intended to simulate the instant movement of the ions. And with this
c  algorithm the program can be easily vectorized and parallelized.
c
c  For each iteration, the percentage of acceptance (POA) is calculated. If the
c  POA is larger than the set percentage (in the input file), the range is
c  reduced by the amount that exceeds that percentage. If it is smaller than
c  that percentage, then the range is increased by the amount needed to make up
c  that percentage.
c
c  This program is for calculating ionic crystals, whose experimental value of
c  perfect lattice spacings are likely known. In the input file, the known
c  lattice spacing is specified as the initial cation-anion distance. If the
c  experiment value is not available, zero (0) should be specified. The
c  program will make a one-dimentional estimate and use that estimate as the
c  initial cation-anion distance.
c
c  This version of the program takes into account the electronic polarizations.
c  This is done in two stages. The first stage gives an estimate of the ions
c  positions with the polarization neglected (to save time). In the second
c  stage, the polarization is included.
c
      include 'common.mc'
c
      real*4 finish(2),start(2),t1(2),t2(2)
c
      data decision/'NO'/
      data ichoose/1, 2, 3, 4, 5,
     1             2, 6, 7, 8, 9,
     2             3, 7,10,11,12,
     3             4, 8,11,13,14,
     4             5, 9,12,14,15/
      data iseed,maxtrial,maxtrialp/-1,10000,5000/
c
      gettime=etime(start)
c
      call getseed
c
      open(unit=1,file='mc.in',status='old')
      open(unit=2,file='mc.res',status='unknown')
c
      call readin
      close(1)
      gettime=etime(t2)
      tother1=t2(1)-start(1)
      tother2=t2(2)-start(2)
c
      gettime=etime(t1)
      call maketable
```

```
      gettime=etime(t2)
      tmaketable1=t2(1)-t1(1)
      tmaketable2=t2(2)-t1(2)
c
      gettime=etime(t1)
      if(ionedim.eq.1)call onedim
      gettime=etime(t2)
      tonedim1=t2(1)-t1(1)
      tonedim2=t2(2)-t1(2)
c
      gettime=etime(t1)
      call lattice
      gettime=etime(t2)
      tlattice1=t2(1)-t1(1)
      tlattice2=t2(2)-t1(2)
c
      gettime=etime(t1)
      call moveion
      if(charge(5).eq.0.0)then
        call total_e
      else
        call total_ec
      endif
      e_moveion=e_total
      gettime=etime(t2)
      tmoveion1=t2(1)-t1(1)
      tmoveion2=t2(2)-t1(2)
c
      gettime=etime(t1)
      call moveionp
      if(charge(5).eq.0.0)then
        call total_e
      else
        call total_ec
      endif
      e_moveionp=e_total
      gettime=etime(t2)
      tmoveionp1=t2(1)-t1(1)
      tmoveionp2=t2(2)-t1(2)
c
      gettime=etime(t1)
      call output
      gettime=etime(t2)
      tother1=tother1+t2(1)-t1(1)
      tother2=tother2+t2(2)-t1(2)
c
      gettime=etime(finish)
c
      finish(1)=finish(1)-start(1)
      finish(2)=finish(2)-start(2)
      start(1)=100.0/finish(1)
      start(2)=100.0/finish(2)
c
      cpu_seconds=tmaketable1
      r_minutes=cpu_seconds/60.0d0
      i_cpu_minutes=r_minutes
      cpu_seconds_remaining=cpu_seconds-i_cpu_minutes*60
      r_hours=r_minutes/60.0d0
      i_cpu_hours=r_hours
      minutes_cpu_remaining=i_cpu_minutes-i_cpu_hours*60
      system_seconds=tmaketable2
      r_minutes=system_seconds/60.0d0
      i_system_minutes=r_minutes
      system_seconds_remaining=system_seconds-i_system_minutes*60
```

```
      r_hours=r_minutes/60.0d0
      i_system_hours=r_hours
      minutes_system_remaining=i_system_minutes-i_system_hours*60
      write(2,2010)tmaketable1,tmaketable2,
     1             tmaketable1*start(1),tmaketable2*start(2),
c    2i_cpu_minutes,cpu_seconds_remaining,
c    3i_system_minutes,system_seconds_remaining,
     4i_cpu_hours,minutes_cpu_remaining,cpu_seconds_remaining,
     5i_system_hours,minutes_system_remaining,system_seconds_remaining
2010  format(///'EXECUTION TIMES:'//
     1t22,'CPU TIME',t51,'SYSTEM TIME'//
     2          'MAKETABLE:',t17,f12.2,' sec',t46,f12.2,
     3                    ' sec',1x,'(',f6.2,'%, ',f6.2,'%)'/
c    4t15,i8,':',f5.2,' min:sec',
c    5t44,i8,':',f5.2,' min:sec'/
     6t14,i6,':',i2,':',f5.2,' hrs:min:sec',
     7t43,i6,':',i2,':',f5.2,' hrs:min:sec'/)
c
      cpu_seconds=tonedim1
      r_minutes=cpu_seconds/60.0d0
      i_cpu_minutes=r_minutes
      cpu_seconds_remaining=cpu_seconds-i_cpu_minutes*60
      r_hours=r_minutes/60.0d0
      i_cpu_hours=r_hours
      minutes_cpu_remaining=i_cpu_minutes-i_cpu_hours*60
      system_seconds=tonedim2
      r_minutes=system_seconds/60.0d0
      i_system_minutes=r_minutes
      system_seconds_remaining=system_seconds-i_system_minutes*60
      r_hours=r_minutes/60.0d0
      i_system_hours=r_hours
      minutes_system_remaining=i_system_minutes-i_system_hours*60
      write(2,2020)tonedim1,tonedim2,
     1tonedim1*start(1),tonedim2*start(2),
c    2i_cpu_minutes,cpu_seconds_remaining,
c    3i_system_minutes,system_seconds_remaining,
     4i_cpu_hours,minutes_cpu_remaining,cpu_seconds_remaining,
     5i_system_hours,minutes_system_remaining,system_seconds_remaining
2020  format('ONEDIM:',
     1t17,f12.2,' sec',t46,f12.2,' sec',1x,'(',f6.2,'%, ',f6.2,'%)'/
c    2t15,i8,':',f5.2,' min:sec',t44,i8,':',f5.2,' min:sec'/
     3t14,i6,':',i2,':',f5.2,' hrs:min:sec',
     4t43,i6,':',i2,':',f5.2,' hrs:min:sec'/)
c
      cpu_seconds=tlattice1
      r_minutes=cpu_seconds/60.0d0
      i_cpu_minutes=r_minutes
      cpu_seconds_remaining=cpu_seconds-i_cpu_minutes*60
      r_hours=r_minutes/60.0d0
      i_cpu_hours=r_hours
      minutes_cpu_remaining=i_cpu_minutes-i_cpu_hours*60
      system_seconds=tlattice2
      r_minutes=system_seconds/60.0d0
      i_system_minutes=r_minutes
      system_seconds_remaining=system_seconds-i_system_minutes*60
      r_hours=r_minutes/60.0d0
      i_system_hours=r_hours
      minutes_system_remaining=i_system_minutes-i_system_hours*60
      write(2,2030)tlattice1,tlattice2,
     1tlattice1*start(1),tlattice2*start(2),
c    2i_cpu_minutes,cpu_seconds_remaining,
c    3i_system_minutes,system_seconds_remaining,
     4i_cpu_hours,minutes_cpu_remaining,cpu_seconds_remaining,
     5i_system_hours,minutes_system_remaining,system_seconds_remaining
```

```
2030  format('LATTICE:',
     1t17,f12.2,' sec',t46,f12.2,' sec',1x,'(',f6.2,'%, ',f6.2,'%)'/
c    2t15,i8,':',f5.2,' min:sec',t44,i8,':',f5.2,' min:sec'/
     3t14,i6,':',i2,':',f5.2,' hrs:min:sec',
     4t43,i6,':',i2,':',f5.2,' hrs:min:sec'/)
c
      cpu_seconds=tmoveion1
      r_minutes=cpu_seconds/60.0d0
      i_cpu_minutes=r_minutes
      cpu_seconds_remaining=cpu_seconds-i_cpu_minutes*60
      r_hours=r_minutes/60.0d0
      i_cpu_hours=r_hours
      minutes_cpu_remaining=i_cpu_minutes-i_cpu_hours*60
      system_seconds=tmoveion2
      r_minutes=system_seconds/60.0d0
      i_system_minutes=r_minutes
      system_seconds_remaining=system_seconds-i_system_minutes*60
      r_hours=r_minutes/60.0d0
      i_system_hours=r_hours
      minutes_system_remaining=i_system_minutes-i_system_hours*60
      write(2,2040)tmoveion1,tmoveion2,
     1tmoveion1*start(1),tmoveion2*start(2),
c    2i_cpu_minutes,cpu_seconds_remaining,
c    3i_system_minutes,system_seconds_remaining,
     4i_cpu_hours,minutes_cpu_remaining,cpu_seconds_remaining,
     5i_system_hours,minutes_system_remaining,system_seconds_remaining
2040  format('MOVEION:',
     1t17,f12.2,' sec',t46,f12.2,' sec',1x,'(',f6.2,'%, ',f6.2,'%)'/
c    2t15,i8,':',f5.2,' min:sec',t44,i8,':',f5.2,' min:sec'/
     3t14,i6,':',i2,':',f5.2,' hrs:min:sec',
     4t43,i6,':',i2,':',f5.2,' hrs:min:sec'/)
c
      cpu_seconds=tmoveionp1
      r_minutes=cpu_seconds/60.0d0
      i_cpu_minutes=r_minutes
      cpu_seconds_remaining=cpu_seconds-i_cpu_minutes*60
      r_hours=r_minutes/60.0d0
      i_cpu_hours=r_hours
      minutes_cpu_remaining=i_cpu_minutes-i_cpu_hours*60
      system_seconds=tmoveionp2
      r_minutes=system_seconds/60.0d0
      i_system_minutes=r_minutes
      system_seconds_remaining=system_seconds-i_system_minutes*60
      r_hours=r_minutes/60.0d0
      i_system_hours=r_hours
      minutes_system_remaining=i_system_minutes-i_system_hours*60
      write(2,2050)tmoveionp1,tmoveionp2,
     1tmoveionp1*start(1),tmoveionp2*start(2),
c    2i_cpu_minutes,cpu_seconds_remaining,
c    3i_system_minutes,system_seconds_remaining,
     4i_cpu_hours,minutes_cpu_remaining,cpu_seconds_remaining,
     5i_system_hours,minutes_system_remaining,system_seconds_remaining
2050  format('MOVEIONP:',
     1t17,f12.2,' sec',t46,f12.2,' sec',1x,'(',f6.2,'%, ',f6.2,'%)'/
c    2t15,i8,':',f5.2,' min:sec',t44,i8,':',f5.2,' min:sec'/
     3t14,i6,':',i2,':',f5.2,' hrs:min:sec',
     4t43,i6,':',i2,':',f5.2,' hrs:min:sec'/)
c
      cpu_seconds=tother1
      r_minutes=cpu_seconds/60.0d0
      i_cpu_minutes=r_minutes
      cpu_seconds_remaining=cpu_seconds-i_cpu_minutes*60
      r_hours=r_minutes/60.0d0
      i_cpu_hours=r_hours
```

```
      minutes_cpu_remaining=i_cpu_minutes-i_cpu_hours*60
      system_seconds=tother2
      r_minutes=system_seconds/60.0d0
      i_system_minutes=r_minutes
      system_seconds_remaining=system_seconds-i_system_minutes*60
      r_hours=r_minutes/60.0d0
      i_system_hours=r_hours
      minutes_system_remaining=i_system_minutes-i_system_hours*60
      write(2,2060)tother1,tother2,tother1*start(1),tother2*start(2),
c    1i_cpu_minutes,cpu_seconds_remaining,
c    2i_system_minutes,system_seconds_remaining,
     3i_cpu_hours,minutes_cpu_remaining,cpu_seconds_remaining,
     4i_system_hours,minutes_system_remaining,system_seconds_remaining
2060  format('OTHER:',
     1t17,f12.2,' sec',t46,f12.2,' sec',1x,'(',f6.2,'%, ',f6.2,'%)'/
c    2t15,i8,':',f5.2,' min:sec',t44,i8,':',f5.2,' min:sec'/
     3t14,i6,':',i2,':',f5.2,' hrs:min:sec',
     4t43,i6,':',i2,':',f5.2,' hrs:min:sec'/)
c
      cpu_seconds=finish(1)
      r_minutes=cpu_seconds/60.0d0
      i_cpu_minutes=r_minutes
      cpu_seconds_remaining=cpu_seconds-i_cpu_minutes*60
      r_hours=r_minutes/60.0d0
      i_cpu_hours=r_hours
      minutes_cpu_remaining=i_cpu_minutes-i_cpu_hours*60
      system_seconds=finish(2)
      r_minutes=system_seconds/60.0d0
      i_system_minutes=r_minutes
      system_seconds_remaining=system_seconds-i_system_minutes*60
      r_hours=r_minutes/60.0d0
      i_system_hours=r_hours
      minutes_system_remaining=i_system_minutes-i_system_hours*60
      write(2,2070)finish(1),finish(2),100.0,100.0,
c    1i_cpu_minutes,cpu_seconds_remaining,
c    2i_system_minutes,system_seconds_remaining,
     3i_cpu_hours,minutes_cpu_remaining,cpu_seconds_remaining,
     4i_system_hours,minutes_system_remaining,system_seconds_remaining
2070  format('TOTAL:',
     1t17,f12.2,' sec',t46,f12.2,' sec',1x,'(',f6.2,'%, ',f6.2,'%)'/
c    2t15,i8,':',f5.2,' min:sec',t44,i8,':',f5.2,' min:sec'/
     3t14,i6,':',i2,':',f5.2,' hrs:min:sec',
     4t43,i6,':',i2,':',f5.2,' hrs:min:sec')
c
      all_seconds=finish(1)+finish(2)
      r_minutes=all_seconds/60.0d0
      minutes=r_minutes
      all_seconds_remaining=all_seconds-minutes*60
      r_hours=r_minutes/60.0d0
      i_hours=r_hours
      minutes_remaining=minutes-i_hours*60
      write(2,2080)all_seconds,
     1               i_hours,minutes_remaining,all_seconds_remaining
2080  format(//'TOTAL TIME:',
     1t14,f12.2,' sec = ',i7,' hrs : ',i2,' min : ',f5.2,' sec')
c
      end
```

```
      SUBROUTINE CASECHANGE(CHA,I)
c
      character*80 cha
c
      do 10 k=1,80
        if(cha(k:k).eq.'a')cha(k:k)='A'
        if(cha(k:k).eq.'b')cha(k:k)='B'
        if(cha(k:k).eq.'c')cha(k:k)='C'
        if(cha(k:k).eq.'d')cha(k:k)='D'
        if(cha(k:k).eq.'e')cha(k:k)='E'
        if(cha(k:k).eq.'f')cha(k:k)='F'
        if(cha(k:k).eq.'g')cha(k:k)='G'
        if(cha(k:k).eq.'h')cha(k:k)='H'
        if(cha(k:k).eq.'i')cha(k:k)='I'
        if(cha(k:k).eq.'j')cha(k:k)='J'
        if(cha(k:k).eq.'k')cha(k:k)='K'
        if(cha(k:k).eq.'l')cha(k:k)='L'
        if(cha(k:k).eq.'m')cha(k:k)='M'
        if(cha(k:k).eq.'n')cha(k:k)='N'
        if(cha(k:k).eq.'o')cha(k:k)='O'
        if(cha(k:k).eq.'p')cha(k:k)='P'
        if(cha(k:k).eq.'q')cha(k:k)='Q'
        if(cha(k:k).eq.'r')cha(k:k)='R'
        if(cha(k:k).eq.'s')cha(k:k)='S'
        if(cha(k:k).eq.'t')cha(k:k)='T'
        if(cha(k:k).eq.'u')cha(k:k)='U'
        if(cha(k:k).eq.'v')cha(k:k)='V'
        if(cha(k:k).eq.'w')cha(k:k)='W'
        if(cha(k:k).eq.'x')cha(k:k)='X'
        if(cha(k:k).eq.'y')cha(k:k)='Y'
        if(cha(k:k).eq.'z')cha(k:k)='Z'
        if(cha(k:k).eq.'(')i=k
10    continue
c
      i=i+1
c
      return
c
      end
```

```
      SUBROUTINE COUNT(CHA,LAST)
c
      character*80 cha
c
      do 1 k=1,80
      if(cha(k:k).ne.' ')last=k
1     continue
c
      return
c
      end
```

```
      SUBROUTINE CUTNULL(CHA)
c
      character*80 cha,temp
c
      ilst=0
c
      do 10 k=1,80
        if(ilst.eq.0.and.cha(k:k).ne.' ')then
          ilst=k
        endif
        if(cha(k:k).ne.' ')last=k
10    continue
c
      if(ilst.eq.0)last=0
      temp=cha(ilst:last)
      cha=temp
c
      return
c
      end
```

```
      SUBROUTINE E_CHANGE(II)
c
      include 'common.mc'
c
      eold=0.0
      enew=0.0
c
      do 10 jj=1,n_points
        if(jj.eq.ii)go to 10
        dx=x(ii)-x(jj)
        dy=y(ii)-y(jj)
        dz=z(ii)-z(jj)
        sr2=dx*dx+dy*dy+dz*dz
        if(sr2.lt.1.0e+2)then
          kk=sr2*10000+0.5
          sr=sroot(kk)
        elseif(sr2.lt.1.0e+4)then
          kk=sr2*100+0.5
          sr=10.0*sroot(kk)
        elseif(sr2.lt.1.0e+6)then
          kk=sr2+0.5
          sr=100.0*sroot(kk)
        elseif(sr2.lt.1.0e+8)then
          kk=sr2*1.0e-2+0.5
          sr=1000.0*sroot(kk)
        else
          write(2,2010)ii,jj,x(ii),x(jj),dx,y(ii),y(jj),dy,
     1                 z(ii),z(jj),dz,sr2
2010      format(///'Sorry. SROOT is not large enough for E_CHANGE.'//
     1               'i=',i4,' j=',i4/
     2               'x(i)=',f20.8,' x(j)=',f20.8,' dx=',f20.8/
     3               'y(i)=',f20.8,' y(j)=',f20.8,' dy=',f20.8/
     4               'z(i)=',f20.8,' z(j)=',f20.8,' dz=',f20.8/
     5               'sr2=',f20.8)
          stop
        endif
        if(sr.gt.100.0)then
          eold=eold+a(ichoose(kind(ii),kind(jj)))/sr
        else
          kk=sr*10000
          if(kk.eq.0)kk=1
          eold=eold+potential(kk,ichoose(kind(ii),kind(jj)))
        endif
        dx=xtemp(ii)-x(jj)
        dy=ytemp(ii)-y(jj)
        dz=ztemp(ii)-z(jj)
        sr2=dx*dx+dy*dy+dz*dz
        if(sr2.lt.1.0e+2)then
          kk=sr2*10000+0.5
          sr=sroot(kk)
        elseif(sr2.lt.1.0e+4)then
          kk=sr2*100+0.5
          sr=10.0*sroot(kk)
        elseif(sr2.lt.1.0e+6)then
          kk=sr2+0.5
          sr=100.0*sroot(kk)
        elseif(sr2.lt.1.0e+8)then
          kk=sr2*1.0e-2+0.5
          sr=1000.0*sroot(kk)
        else
          write(2,2010)ii,jj,x(ii),x(jj),dx,y(ii),y(jj),dy,
     1                 z(ii),z(jj),dz,sr2
          stop
        endif
```

```
          if(sr.gt.100.0)then
            enew=enew+a(ichoose(kind(ii),kind(jj)))/sr
          else
            kk=sr*10000
            if(kk.eq.0)kk=1
            enew=enew+potential(kk,ichoose(kind(ii),kind(jj)))
          endif
10      continue
c
        delta_e=enew-eold
c
        return
c
        end
```

```
      SUBROUTINE E_CHANGEC(II)
c
      include 'common.mc'
c
      eold=0.0
      enew=0.0
c
      do 20 jj=1,n_points
        if(jj.eq.ii)go to 20
        c5=0.0
        if(kind(ii).gt.10)then
          kind_ii=kind(ii)-10
          if(kind(jj).gt.10)then
            kind_jj=kind(jj)-10
            c5=a(ichoose(5,kind_ii))+a(ichoose(5,kind_jj))+a(15)
          else
            kind_jj=kind(jj)
            c5=a(ichoose(5,kind_jj))
          endif
          go to 10
        else
          kind_ii=kind(ii)
        endif
        if(kind(jj).gt.10)then
          kind_jj=kind(jj)-10
          c5=a(ichoose(5,kind_ii))
        else
          kind_jj=kind(jj)
        endif
10      dx=x(ii)-x(jj)
        dy=y(ii)-y(jj)
        dz=z(ii)-z(jj)
        sr2=dx*dx+dy*dy+dz*dz
        if(sr2.lt.1.0e+2)then
          kk=sr2*10000+0.5
          sro=sroot(kk)
        elseif(sr2.lt.1.0e+4)then
          kk=sr2*100+0.5
          sro=10.0*sroot(kk)
        elseif(sr2.lt.1.0e+6)then
          kk=sr2+0.5
          sro=100.0*sroot(kk)
        elseif(sr2.lt.1.0e+8)then
          kk=sr2*1.0e-2+0.5
          sro=1000.0*sroot(kk)
        else
          write(2,2010)ii,jj,x(ii),x(jj),dx,y(ii),y(jj),dy,
     1                 z(ii),z(jj),dz,sr2
2010      format(///'Sorry. SROOT is not large enough for E_CHANGEC.'//
     1             'i=',i4,' j=',i4/
     2             'x(i)=',f20.8,' x(j)=',f20.8,' dx=',f20.8/
     3             'y(i)=',f20.8,' y(j)=',f20.8,' dy=',f20.8/
     4             'z(i)=',f20.8,' z(j)=',f20.8,' dz=',f20.8/
     5             'sr2=',f20.8)
          stop
        endif
        if(sro.gt.100.0)then
          eold=eold+a(ichoose(kind_ii,kind_jj))/sro
        else
          kk=sro*10000
          if(kk.eq.0)kk=1
          eold=eold+potential(kk,ichoose(kind_ii,kind_jj))
        endif
        dx=xtemp(ii)-x(jj)
```

```
          dy=ytemp(ii)-y(jj)
          dz=ztemp(ii)-z(jj)
          sr2=dx*dx+dy*dy+dz*dz
          if(sr2.lt.1.0e+2)then
            kk=sr2*10000+0.5
            srn=sroot(kk)
          elseif(sr2.lt.1.0e+4)then
            kk=sr2*100+0.5
            srn=10.0*sroot(kk)
          elseif(sr2.lt.1.0e+6)then
            kk=sr2+0.5
            srn=100.0*sroot(kk)
          elseif(sr2.lt.1.0e+8)then
            kk=sr2*1.0e-2+0.5
            srn=1000.0*sroot(kk)
          else
            write(2,2010)ii,jj,x(ii),x(jj),dx,y(ii),y(jj),dy,
     1                   z(ii),z(jj),dz,sr2
            stop
          endif
          if(srn.gt.100.0)then
            enew=enew+a(ichoose(kind_ii,kind_jj))/srn
          else
            kk=srn*10000
            if(kk.eq.0)kk=1
            enew=enew+potential(kk,ichoose(kind_ii,kind_jj))
          endif
          if(c5.ne.0.0)enew=enew+c5/srn-c5/sro
20      continue
c
        delta_e=enew-eold
c
        return
c
        end
```

```
      SUBROUTINE E_CHANGEP(II)
c
      include 'common.mc'
c
      eold=0.0
      enew=0.0
c
      do 10 jj=1,n_points
        if(jj.eq.ii)go to 10
        dx=x(ii)-x(jj)
        dy=y(ii)-y(jj)
        dz=z(ii)-z(jj)
        sr2=dx*dx+dy*dy+dz*dz
        if(sr2.lt.1.0e+2)then
          kk=sr2*10000+0.5
          sr=sroot(kk)
        elseif(sr2.lt.1.0e+4)then
          kk=sr2*100+0.5
          sr=10.0*sroot(kk)
        elseif(sr2.lt.1.0e+6)then
          kk=sr2+0.5
          sr=100.0*sroot(kk)
        elseif(sr2.lt.1.0e+8)then
          kk=sr2*1.0e-2+0.5
          sr=1000.0*sroot(kk)
        else
          write(2,2010)ii,jj,x(ii),x(jj),dx,y(ii),y(jj),dy,
     1               z(ii),z(jj),dz,sr2
2010      format(///'Sorry. SROOT is not large enough for E_CHANGEP.'//
     1              'i=',i4,' j=',i4/
     2              'x(i)=',f20.8,' x(j)=',f20.8,' dx=',f20.8/
     3              'y(i)=',f20.8,' y(j)=',f20.8,' dy=',f20.8/
     4              'z(i)=',f20.8,' z(j)=',f20.8,' dz=',f20.8/
     5              'sr2=',f20.8)
          stop
        endif
        sr3=sr*sr2
        uvx=dx/sr
        uvy=dy/sr
        uvz=dz/sr
        pi_dot_pj=p(1,ii)*p(1,jj)+p(2,ii)*p(2,jj)+p(3,ii)*p(3,jj)
        pi_dot_rij=p(1,ii)*uvx+p(2,ii)*uvy+p(3,ii)*uvz
        pj_dot_rij=p(1,jj)*uvx+p(2,jj)*uvy+p(3,jj)*uvz
        w=(pi_dot_pj-3.0*pi_dot_rij*pj_dot_rij)/sr3
        if(sr.gt.100.0)then
          eold=eold+a(ichoose(kind(ii),kind(jj)))/sr+w
        else
          kk=sr*10000
          if(kk.eq.0)kk=1
          eold=eold+potential(kk,ichoose(kind(ii),kind(jj)))+w
        endif
        dx=xtemp(ii)-x(jj)
        dy=ytemp(ii)-y(jj)
        dz=ztemp(ii)-z(jj)
        sr2=dx*dx+dy*dy+dz*dz
        if(sr2.lt.1.0e+2)then
          kk=sr2*10000+0.5
          sr=sroot(kk)
        elseif(sr2.lt.1.0e+4)then
          kk=sr2*100+0.5
          sr=10.0*sroot(kk)
        elseif(sr2.lt.1.0e+6)then
          kk=sr2+0.5
          sr=100.0*sroot(kk)
```

```
        elseif(sr2.lt.1.0e+8)then
          kk=sr2*1.0e-2+0.5
          sr=1000.0*sroot(kk)
        else
          write(2,2010)ii,jj,x(ii),x(jj),dx,y(ii),y(jj),dy,
     1                 z(ii),z(jj),dz,sr2
          stop
        endif
        sr3=sr*sr2
        uvx=dx/sr
        uvy=dy/sr
        uvz=dz/sr
        pi_dot_rij=p(1,ii)*uvx+p(2,ii)*uvy+p(3,ii)*uvz
        pj_dot_rij=p(1,jj)*uvx+p(2,jj)*uvy+p(3,jj)*uvz
        w=(pi_dot_pj-3.0*pi_dot_rij*pj_dot_rij)/sr3
        if(sr.gt.100.0)then
          enew=enew+a(ichoose(kind(ii),kind(jj)))/sr+w
        else
          kk=sr*10000
          if(kk.eq.0)kk=1
          enew=enew+potential(kk,ichoose(kind(ii),kind(jj)))+w
        endif
10    continue
c
      delta_e=enew-eold
c
      return
c
      end
```

```
      SUBROUTINE E_CHANGEPC(II)
c
      include 'common.mc'
c
      eold=0.0
      enew=0.0
c
      do 20 jj=1,n_points
        if(jj.eq.ii)go to 20
        c5=0.0
        if(kind(ii).gt.10)then
          kind_ii=kind(ii)-10
          if(kind(jj).gt.10)then
            kind_jj=kind(jj)-10
            c5=a(ichoose(5,kind_ii))+a(ichoose(5,kind_jj))+a(15)
          else
            kind_jj=kind(jj)
            c5=a(ichoose(5,kind_jj))
          endif
          go to 10
        else
          kind_ii=kind(ii)
        endif
        if(kind(jj).gt.10)then
          kind_jj=kind(jj)-10
          c5=a(ichoose(5,kind_ii))
        else
          kind_jj=kind(jj)
        endif
10      dx=x(ii)-x(jj)
        dy=y(ii)-y(jj)
        dz=z(ii)-z(jj)
        sr2=dx*dx+dy*dy+dz*dz
        if(sr2.lt.1.0e+2)then
          kk=sr2*10000+0.5
          sro=sroot(kk)
        elseif(sr2.lt.1.0e+4)then
          kk=sr2*100+0.5
          sro=10.0*sroot(kk)
        elseif(sr2.lt.1.0e+6)then
          kk=sr2+0.5
          sro=100.0*sroot(kk)
        elseif(sr2.lt.1.0e+8)then
          kk=sr2*1.0e-2+0.5
          sro=1000.0*sroot(kk)
        else
          write(2,2010)ii,jj,x(ii),x(jj),dx,y(ii),y(jj),dy,
     1               z(ii),z(jj),dz,sr2
 2010     format(///'Sorry. SROOT is not large enough for E_CHANGEPC.'//
     1               'i=',i4,' j=',i4/
     2               'x(i)=',f20.8,' x(j)=',f20.8,' dx=',f20.8/
     3               'y(i)=',f20.8,' y(j)=',f20.8,' dy=',f20.8/
     4               'z(i)=',f20.8,' z(j)=',f20.8,' dz=',f20.8/
     5               'sr2=',f20.8)
          stop
        endif
        sr3=sro*sr2
        uvx=dx/sro
        uvy=dy/sro
        uvz=dz/sro
        pi_dot_pj=p(1,ii)*p(1,jj)+p(2,ii)*p(2,jj)+p(3,ii)*p(3,jj)
        pi_dot_rij=p(1,ii)*uvx+p(2,ii)*uvy+p(3,ii)*uvz
        pj_dot_rij=p(1,jj)*uvx+p(2,jj)*uvy+p(3,jj)*uvz
        w=(pi_dot_pj-3.0*pi_dot_rij*pj_dot_rij)/sr3
```

```
        if(sro.gt.100.0)then
          eold=eold+a(ichoose(kind_ii,kind_jj))/sro+w
        else
          kk=sro*10000
          if(kk.eq.0)kk=1
          eold=eold+potential(kk,ichoose(kind_ii,kind_jj))+w
        endif
        dx=xtemp(ii)-x(jj)
        dy=ytemp(ii)-y(jj)
        dz=ztemp(ii)-z(jj)
        sr2=dx*dx+dy*dy+dz*dz
        if(sr2.lt.1.0e+2)then
          kk=sr2*10000+0.5
          srn=sroot(kk)
        elseif(sr2.lt.1.0e+4)then
          kk=sr2*100+0.5
          srn=10.0*sroot(kk)
        elseif(sr2.lt.1.0e+6)then
          kk=sr2+0.5
          srn=100.0*sroot(kk)
        elseif(sr2.lt.1.0e+8)then
          kk=sr2*1.0e-2+0.5
          srn=1000.0*sroot(kk)
        else
          write(2,2010)ii,jj,x(ii),x(jj),dx,y(ii),y(jj),dy,
     1                 z(ii),z(jj),dz,sr2
          stop
        endif
        sr3=srn*sr2
        uvx=dx/srn
        uvy=dy/srn
        uvz=dz/srn
        pi_dot_rij=p(1,ii)*uvx+p(2,ii)*uvy+p(3,ii)*uvz
        pj_dot_rij=p(1,jj)*uvx+p(2,jj)*uvy+p(3,jj)*uvz
        w=(pi_dot_pj-3.0*pi_dot_rij*pj_dot_rij)/sr3
        if(srn.gt.100.0)then
          enew=enew+a(ichoose(kind_ii,kind_jj))/srn+w
        else
          kk=srn*10000
          if(kk.eq.0)kk=1
          enew=enew+potential(kk,ichoose(kind_ii,kind_jj))+w
        endif
        if(c5.ne.0.0)enew=enew+c5/srn-c5/sro
20    continue
c
      delta_e=enew-eold
c
      return
c
      end
```

```
      subroutine getp
c
      include 'common.mc'

      data pi4over3/4.1887902/
c
      do 30 ii=1,n_points
        xp=0.0
        yp=0.0
        zp=0.0
        do 20 jj=ii+1,n_points
          dx=x(ii)-x(jj)
          dy=y(ii)-y(jj)
          dz=z(ii)-z(jj)
          sr2=dx*dx+dy*dy+dz*dz
          if(sr2.lt.1.0e+2)then
            kk=sr2*10000+0.5
            sr=sroot(kk)
          elseif(sr2.lt.1.0e+4)then
            kk=sr2*100+0.5
            sr=10.0*sroot(kk)
          elseif(sr2.lt.1.0e+6)then
            kk=sr2+0.5
            sr=100.0*sroot(kk)
          elseif(sr2.lt.1.0e+8)then
            kk=sr2*1.0e-2+0.5
            sr=1000.0*sroot(kk)
          else
            write(2,2010)ii,jj,x(ii),x(jj),dx,y(ii),y(jj),dy,
     1                   z(ii),z(jj),dz,sr2
 2010       format(///'Sorry. SROOT is not large enough for GETP.'//
     1             'i=',i4,' j=',i4/
     2             'x(i)=',f20.8,' x(j)=',f20.8,' dx=',f20.8/
     3             'y(i)=',f20.8,' y(j)=',f20.8,' dy=',f20.8/
     4             'z(i)=',f20.8,' z(j)=',f20.8,' dz=',f20.8/
     5             'sr2=',f20.8)
            stop
          endif
          uvx=dx/sr
          uvy=dy/sr
          uvz=dz/sr
          sr3=sr*sr2
          sr5=sr2*sr3
          sr6=sr3*sr3
          if(kind(ii).gt.10)then
            kind_ii=kind(ii)-10
          else
            kind_ii=kind(ii)
          endif
          if(kind(jj).gt.10)then
            kind_jj=kind(jj)-10
          else
            kind_jj=kind(jj)
          endif
          p_plus=alpha(kind_ii)*charge(kind_jj)/sr3
          alpha_product=alpha(kind_ii)*alpha(kind_jj)
          p_minus=-2.0*alpha_product*charge(kind_ii)/sr5
          f=4.0*alpha_product/sr6
          pt_plus=p_plus
          pt_minus=p_minus
          do 10 kk=1,6
            p_plus=f*p_plus
            p_minus=f*p_minus
            pt_plus=pt_plus+p_plus
```

```
          pt_minus=pt_minus+p_minus
10      continue
        pt=pt_plus+pt_minus
        xp=xp+pt*uvx
        yp=yp+pt*uvy
        zp=zp+pt*uvz
20    continue
      p(1,ii)=xp
      p(2,ii)=yp
      p(3,ii)=zp
30  continue
c
    return
c
    end
```

```fortran
      subroutine getseed
c
      include 'common.mc'
c
      ia1=7141
      ia2=8121
      ia3=4561
      ic1=54773
      ic2=28411
      ic3=51349
      m1=259200
      m2=143356
      m3=243000
      negone=-1
      rm11=1.0/m1
      rm12=2.0/m1
      rm2=1.0/m2
      rm3=1.0/m3
c
      read(5,5010)i,j,k
5010  format(t12,i2,t15,i2,t18,i2)
c
      if(i.eq.0)i=333
      if(j.eq.0)j=33
      if(k.eq.0)k=3
c
      iseed=real(i)*real(k)/real(j)
c
      read(5,5020,end=10)seed
5020  format(e80.0)
      iseed=abs(seed)
c
10    if(iseed.eq.0)iseed=3333
c
      ix1=mod(ic1+iseed,m1)
      ix1=mod(ia1*ix1+ic1,m1)
      ix2=mod(ix1,m2)
      ix1=mod(ia1*ix1+ic1,m1)
      ix3=mod(ix1,m3)
      do 20 k=1,97
        ix1=mod(ia1*ix1+ic1,m1)
        ix2=mod(ia2*ix2+ic2,m2)
        r=ix1+ix2*rm2
        random1(k)=r*rm11
        random2(k)=r*rm12+negone
20    continue
c
      return
c
      end
```

```
      SUBROUTINE LATTICE
c
      include 'common.mc'
c
      character*80 cha
c
      n_per_sideml=n_per_side-1
      n_per_layer=n_per_side*n_per_side
      n_points=n_per_layer*n_per_side
      if(mod(n_points,2).eq.0)then
        ieven=1
      else
        ieven=-1
      endif
      n_points_ml=n_points-1
      n_points_m2=n_points-2
      icenter=n_per_side/2-1+mod(n_per_side,2)
      icenter=icenter*(n_per_layer+n_per_side+1)+1
      myfront=icenter+1
      myback=icenter-1
      myleft=icenter-n_per_side
      myright=icenter+n_per_side
      myabove=icenter+n_per_layer
      mybelow=icenter-n_per_layer
c
      if(kind_of_center.eq.1)then
        kind_of_next=2
      else
        kind_of_next=1
      endif
c
      if(mod(icenter,2).eq.0)then
        sign=1.0
      else
        sign=-1.0
      endif
c
      k=1
      do 30 iz=0,n_per_sideml
        do 20 iy=0,n_per_sideml
          do 10 ix=0,n_per_sideml
            x(k)=spacing*ix
            y(k)=spacing*iy
            z(k)=spacing*iz
            if(sign.gt.0.0)then
              kind(k)=kind_of_next
            else
              kind(k)=kind_of_center
            endif
            iorder(k)=k
            all_charge=all_charge+charge(kind(k))
            sign=-sign
            if(ieven.eq.1)then
              if(mod(k,n_per_side).eq.0)sign=-sign
              if(mod(k,n_per_layer).eq.0)sign=-sign
            endif
            k=k+1
 10       continue
 20     continue
 30   continue
c
      do 40 k=3,n_kinds
        km2=k-2
        cha=defect_site(km2)(1:1)
```

```
          if(cha.eq.'C')then
            center_ion=ion(k)
            all_charge=all_charge-charge(kind(icenter))
     1                      +charge(k)
            kind(icenter)=k
            go to 40
          endif
          if(defect_site(km2)(1:1).eq.'F')then
            if(myfront.lt.0.or.myfront.gt.n_points)then
              write(6,6010)
6010          format(///'Sorry. The FRONT position is not available.'///)
              stop
            endif
            all_charge=all_charge-charge(kind(myfront))
     1                      +charge(k)
            kind(myfront)=k
            go to 40
          endif
          if(defect_site(km2)(1:1).eq.'L')then
            if(myleft.lt.0.or.myleft.gt.n_points)then
              write(6,6030)
6030          format(///'Sorry. The LEFT position is not available.'///)
              stop
            endif
            all_charge=all_charge-charge(kind(myleft))
     1                      +charge(k)
            kind(myleft)=k
            go to 40
          endif
          if(defect_site(km2)(1:1).eq.'R')then
            if(myright.lt.0.or.myright.gt.n_points)then
              write(6,6040)
6040          format(///'Sorry. The RIGHT position is not available.'///)
              stop
            endif
            all_charge=all_charge-charge(kind(myright))
     1                      +charge(k)
            kind(myright)=k
            go to 40
          endif
          if(defect_site(km2)(1:1).eq.'A')then
            if(myabove.lt.0.or.myabove.gt.n_points)then
              write(6,6050)
6050          format(///'Sorry. The ABOVE position is not available.'///)
              stop
            endif
            all_charge=all_charge-charge(kind(myabove))
     1                      +charge(k)
            kind(myabove)=k
            go to 40
          endif
          if(defect_site(km2)(1:2).eq.'BA')then
            if(myback.lt.0.or.myback.gt.n_points)then
              write(6,6020)
6020          format(///'Sorry. The BACK position is not available.'///)
              stop
            endif
            all_charge=all_charge-charge(kind(myback))
     1                      +charge(k)
            kind(myback)=k
            go to 40
          endif
          if(defect_site(km2)(1:2).eq.'BE')then
            if(mybelow.lt.0.or.mybelow.gt.n_points)then
```

```
          write(6,6060)
6060      format(///'Sorry. The BELOW position is not available.'///)
          stop
        endif
        all_charge=all_charge-charge(kind(mybelow))
     1                  +charge(k)
        kind(mybelow)=k
        go to 40
      endif
      write(6,6070)
6070  format(///
     1     'Sorry. One of the positions of the defect ions is wrong.'
     2     //'The correct positions are:'//
     3     'CENTRAL, FRONT, BACK, LEFT, RIGHT, ABOVE, and BELOW.'//
     4     'These positions are with respect to the central ion.'///)
      stop
40    continue
c
      if(all_charge.eq.0.0)go to 90
c
      n_per_sidem2=n_per_side-2
      n_face_ions=n_points-n_per_sidem2*n_per_sidem2*n_per_sidem2
      charge(5)=-all_charge/n_face_ions
      all_charge=all_charge+n_face_ions*charge(5)
      k=1
      do 70 iz=0,n_per_siderml
        do 60 iy=0,n_per_sidem1
          do 50 ix=0,n_per_sidem1
            if(ix.eq.0.or.iy.eq.0.or.iz.eq.0.or.
     1         ix.eq.n_per_sidem1.or.
     2         iy.eq.n_per_sidem1.or.
     3         iz.eq.n_per_sidem1)kind(k)=kind(k)+10
            k=k+1
50        continue
60      continue
70    continue
      do 80 j=1,5
        a(ichoose(5,j))=charge(5)*charge(j)
80    continue
c
90    dum=x(icenter)
      x(icenter)=x(n_points)
      x(n_points)=dum
      dum=y(icenter)
      y(icenter)=y(n_points)
      y(n_points)=dum
      dum=z(icenter)
      z(icenter)=z(n_points)
      z(n_points)=dum
      idum=kind(icenter)
      kind(icenter)=kind(n_points)
      kind(n_points)=idum
c
      return
c
      end
```

```
      SUBROUTINE MAKETABLE
c
      include 'common.mc'
c
      real*8 r,r2,r6
c
      r=1.0d-4
      do 20 i=1,1000000
        sroot(i)=sqrt(r)
        do 10 j=1,10
          if(a(j).eq.0.0)go to 10
          if(b(j).eq.0.0)then
            if(r.lt.d(j))then
              potential(i,j)=9.99d99
            else
              potential(i,j)=a(j)/r
            endif
          else
            r2=r*r
            r6=r2*r2*r2
            if(r.lt.d(j))then
              potential(i,j)=9.99d99
            else
              potential(i,j)=a(j)/r+b(j)*exp(r*rho(j))-c(j)/r6
            endif
          endif
 10     continue
        r=r+1.0d-4
 20   continue
c
      return
c
      end
```

```
      subroutine moveion
c
      include 'common.mc'
c
      range=0.1*spacing
      m=mod(100,n_points_ml)
      if(m.eq.0)then
        n100=100/n_points_ml
      else
        n100=100.0/n_points_ml+1
      endif
      trials=n100*n_points_ml
c
10    continue
c
      do 40 i=1,n_points_ml
c
        xtemp(iorder(i))=x(iorder(i))+range*ran2(dum)
c
        if(iorder(i).eq.myabove)then
          ytemp(iorder(i))=y(iorder(i))
          go to 20
        endif
c
        ytemp(iorder(i))=y(iorder(i))+range*ran2(dum)
c
        if(iorder(i).eq.myfront)then
          ztemp(iorder(i))=z(iorder(i))
          go to 30
        endif
c
20      ztemp(iorder(i))=z(iorder(i))+range*ran2(dum)
c
30      if(charge(5).eq.0.0)then
          call e_change(iorder(i))
        else
          call e_changec(iorder(i))
        endif
c
        if(delta_e.ge.0.0d0)then
          xtemp(iorder(i))=x(iorder(i))
          ytemp(iorder(i))=y(iorder(i))
          ztemp(iorder(i))=z(iorder(i))
          go to 40
        endif
c
        move(iorder(i))=move(iorder(i))+1
        totalmove=totalmove+1
c
40    continue
c
      do 50 i=1,n_points_ml
        x(i)=xtemp(i)
        y(i)=ytemp(i)
        z(i)=ztemp(i)
50    continue
c
      n_trials=n_trials+1
      if(n_trials.eq.maxtrial)return
      if(mod(n_trials,n100).eq.0)then
        poa=totalmove/trials
        amount=poa+pre_poa
        range=(1+amount)*range
        if(range.lt.accuracy)return
```

```
          move_total=move_total+totalmove
          totalmove=0
        endif
        call shuffle
        go to 10
c
        end
```

```
      subroutine moveionp
c
      include 'common.mc'
c
      range=0.3
c
10    call getp
c
      do 40 i=1,n_points_m1
c
        xtemp(iorder(i))=x(iorder(i))+range*ran2(dum)
c
        if(iorder(i).eq.myabove)then
          ytemp(iorder(i))=y(iorder(i))
          go to 20
        endif
c
        ytemp(iorder(i))=y(iorder(i))+range*ran2(dum)
c
        if(iorder(i).eq.myfront)then
          ztemp(iorder(i))=z(iorder(i))
          go to 30
        endif
c
20      ztemp(iorder(i))=z(iorder(i))+range*ran2(dum)
c
30      if(charge(5).eq.0.0)then
          call e_changep(iorder(i))
        else
          call e_changepc(iorder(i))
        endif
c
        if(delta_e.ge.0.0d0)then
          xtemp(iorder(i))=x(iorder(i))
          ytemp(iorder(i))=y(iorder(i))
          ztemp(iorder(i))=z(iorder(i))
          go to 40
        endif
c
        move(iorder(i))=move(iorder(i))+1
        totalmove=totalmove+1
c
40    continue
c
      do 50 i=1,n_points_m1
        x(i)=xtemp(i)
        y(i)=ytemp(i)
        z(i)=ztemp(i)
50    continue
c
      n_trialsp=n_trialsp+1
      if(n_trialsp.eq.maxtrialp)return
      if(mod(n_trialsp,n100).eq.0)then
        poa=totalmove/trials
        amount=poa+pre_poa
        range=(1+amount)*range
        if(range.lt.accuracy)return
        move_totalp=move_totalp+totalmove
        totalmove=0
      endif
      call shuffle
      go to 10
c
      end
```

```
      SUBROUTINE NUMBER(CHA,IOR,REAL,INTEGER)
c
      include 'common.mc'
c
      character*80 cha,char
      integer*4 isign(2),nornend(50)
      real*8 after,before,real,sign
c
      data chn/'0','1','2','3','4','5','6','7','8','9','10','11','12',
     1'13','14','15','16','17','18','19','20','21','22','23','24','25',
     2'26','27','28','29','30','31','32','33','34','35','36','37','38',
     3'39','40','41','42','43','44','45','46','47','48','49','50','51',
     4'52','53','54','55','56','57','58','59','60','61','62','63','64',
     5'65','66','67','68','69','70','71','72','73','74','75','76','77',
     6'78','79','80','81','82','83','84','85','86','87','88','89','90',
     7'91','92','93','94','95','96','97','98','99'/
c
      data norn/'1st','2nd','3rd','4th','5th','6th','7th','8th','9th',
     1'10th','11th','12th','13th','14th','15th','16th','17th','18th',
     2'19th','20th','21st','22nd','23rd','24th','25th','26th','27th',
     3'28th','29th','30th','31st','32nd','33rd','34th','35th','36th',
     4'37th','38th','39th','40th','41st','42nd','43rd','44th','45th',
     5'46th','47th','48th','49th','50th'/
c
      data nornend/3,3,3,3,3,3,3,3,3,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,
     14,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4/
c
c DETERMINE WHERE THE LAST CHARACTER IS.
c
      do 1 k=1,80
        if(cha(k:k).ne.' ')last=k
1     continue
c
c THE ALLOWED CHARACTERS ARE NUMBERS, PLUS OR MINUS SIGNS, THE DECIMAL POINT,
c AND THE LETTERS E(e) AND D(d).
c
      idore=0
      ipoint=0
      isign(1)=0
      isign(2)=0
      ndore=0
      npoint=0
      nsign=0
c
      do 4 k=1,last
c
      if(cha(k:k).eq.'+'.or.cha(k:k).eq.'-')then
      nsign=nsign+1
      if(nsign.gt.2)then
      write(6,101)
101   format(////////////////////////)
      write(6,201)cha(1:last),nsign
201   format('Sorry. You have to stop here.'///'In the number you input:
     1',a,','//'there are at least ',i2,' signs. Only two signs are all
     2owed.'//'Stop.'/)
      stop
      endif
      isign(nsign)=k
      go to 4
      endif
c
      if(cha(k:k).eq.'D'.or.cha(k:k).eq.'d'.or.cha(k:k).eq.'E'
     1   .or.cha(k:k).eq.'e')then
      ndore=ndore+1
```

```
      if(ndore.gt.1)then
      write(6,101)
      write(6,202)cha(1:last),ndore
202   format('Sorry. You have to stop here.'///'In the number you input:
     1',a,','//'there are at least ',i1,' D(d)s or E(e)s. Only one D(d)
     2or E(e) is allowed.'//'Stop.'/)
      stop
      endif
      idore=k
      go to 4
      endif
c
      if(cha(k:k).eq.'.')then
      npoint=npoint+1
      if(npoint.gt.1)then
      write(6,101)
      write(6,203)cha(1:last),npoint
203   format('Sorry. You have to stop here.'///'In the number you input:
     1 ',a,','//'there are at least ',i2,' decimal points. This is not a
     2llowed.'//'Stop.'/)
      stop
      endif
      ipoint=k
      go to 4
      endif
c
      do 3 l=1,10
        if(cha(k:k).eq.chn(l))go to 4
3     continue
c
      write(6,101)
      write(6,204)cha(1:last),norn(k)(1:nornend(k))
204   format('Sorry. You have to stop here.'///'In the number you input:
     1 ',a,','//'the ',a,' character is not allowed.'///
     2'The allowed characters are:'//
     3'              Numbers'/
     4'              Plus or minus signs'/
     5'              The decimal point'/
     6'              The letters D(d) and E(e)'///'Stop.'/)
      stop
c
4     continue
c
c SEE WHERE THE FIRST SIGN IS IF THERE IS ONE.
c
      if(isign(1).lt.2)go to 5
      if(isign(1)-1.eq.idore)go to 5
      write(6,101)
      write(6,205)cha(1:last)
205   format('Sorry. You have to stop here.'///'In the number you input:
     1',a,','//'the first sign is not the first character.'//'And it is
     2not the one after the letter D(d) or E(e) either.'//'This is not
     3allowed. Stop.'/)
      stop
c
c SEE WHERE THE SECOND SIGN IS IF THERE IS ONE.
c
5     if(isign(2).eq.0)go to 6
      if(isign(2)-1.eq.idore)go to 6
      write(6,101)
      write(6,206)cha(1:last)
206   format('Sorry. You have to stop here.'///'In the number you input:
     1',a,','//'the second sign is not right after the letter D(d) or E
     2(e).'//'This is not allowed. Stop.'/)
```

```
      stop
c
c SEE WHERE THE DECIMAL POINT IS IF THERE IS ONE.
c
6     if(ipoint.eq.0.or.idore.eq.0)go to 7
      if(ipoint.lt.idore)go to 7
      write(6,101)
      write(6,207)cha(1:last)
207   format('Sorry. You have to stop here.'///'In the number you input:
     1',a,','//'the decimal point is after the letter D(d) or E(e).'//'
     2This is not allowed. Stop.'/)
      stop
c
c THE LETTER D(d) OR E(e) CAN NOT BE THE FIRST CHARACTER.
c
7     if(idore.eq.1)then
      write(6,101)
      write(6,208)cha(1:last)
208   format('Sorry. You have to stop here.'///'In the number you input:
     1',a,','//'the first character is the letter D(d) or E(e).'//'This
     2is not allowed. Stop.'/)
      stop
      endif
c
c GET THE NUMBER.
c
      after=0.0d0
      before=0.0d0
      ndecimal=0
      nexp=0
c
c GET THE PART OF THE NUMBER BEFORE THE DECIMAL POINT.
c
      sign=1.0d0
      ibegin=1
c
      if(ipoint.eq.1)go to 14
c
      if(isign(1).eq.0)go to 8
c
      if(cha(1:1).eq.'-')sign=-1.0d0
c
      if(isign(1).eq.1)ibegin=2
c
8     if(ipoint.ne.0)then
      iend=ipoint-1
      go to 9
      endif
c
      if(idore.eq.0)then
      iend=last
      else
      iend=idore-1
      endif
c
9     do 13 k=ibegin,iend
      do 12 l=1,10
      if(cha(k:k).eq.chn(l))then
      if(l.eq.1)go to 11
      real=1.0d0
      do 10 m=1,iend-k
      real=real*10
10    continue
11    before=before+real*(l-1)
```

```
      go to 13
      endif
12    continue
13    continue
c
c GET THE PART OF THE NUMBER AFTER THE DECIMAL POINT.
c
14    if(ipoint.eq.last.or.ipoint+1.eq.idore.or.ipoint.eq.0)go to 19
c
      iend=idore-1
      if(idore.eq.0)iend=last
      ndecimal=iend-ipoint
c
      do 18 k=ipoint+1,iend
      do 17 l=1,10
      if(cha(k:k).eq.chn(l))then
      if(l.eq.1)go to 16
      real=1.0d0
      do 15 m=1,iend-k
      real=real*10
15    continue
16    after=after+real*(l-1)
      go to 18
      endif
17    continue
18    continue
c
c GET THE PART OF THE NUMBER AFTER THE LETTER D(d) OR E(e).
c
19    if(idore.eq.0.or.idore.eq.last)go to 25
c
      nexpsign=1
      ibegin=idore+1
c
      if(isign(2).eq.0)then
      if(isign(1).lt.2)go to 20
      if(cha(isign(1):isign(1)).eq.'-')nexpsign=-1
      else
      if(cha(isign(2):isign(2)).eq.'-')nexpsign=-1
      endif
c
      ibegin=idore+2
c
20    do 24 k=ibegin,last
      do 23 l=1,10
      if(cha(k:k).eq.chn(l))then
      if(l.eq.1)go to 22
      n=1
      do 21 m=1,last-k
      n=n*10
21    continue
22    nexp=nexp+n*(l-1)
      go to 24
      endif
23    continue
24    continue
c
c GET THE WHOLE NUMBER TOGETHER.
c
25    do 26 k=1,ndecimal
        before=before*10
26    continue
c
      real=after+before
```

```
c
      ndecimal=nexpsign*nexp-ndecimal
c
      do 27 k=1,iabs(ndecimal)
      if(ndecimal.gt.0)then
      real=real*10
      else
      real=real/10
      endif
27    continue
c
      real=sign*real
c
      if(ior.ne.1)return
c
c THE INTEGER VALUE SHOULD BE IN THE RANGE -2147483648 TO 2147483647.
c
      if(real.lt.-2147483648.0d0.or.real.gt.2147483647.0d0)go to 28
c
      integer=int(real)
      return
c
28    write(6,101)
      write(6,209)cha(1:last)
209   format('Warning!!! There is an integer out of range.'//
     1'The allowed integer range in this FORTRAN is -2147483647 to 21474
     283647.'////'Your number is ',a,'. It is out of this range.'////
     3'The number will be set to the maximum or minimum according to the
     4 sign.'///)
c
      if(sign.gt.0.0d0)then
        integer=2147483647
      else
        integer=-2147483647
      endif
c
      return
c
      end
```

```
      subroutine onedim
c
      include 'common.mc'
c
      spacing=10.0
      step=-1.0
      sign=1.0
c
      do 10 k=1,10
        x(k)=(k-1)*spacing
        if(sign.gt.0.0)then
          kind(k)=1
        else
          kind(k)=2
        endif
        sign=-sign
10    continue
c
      call total_el
      eold=e_total
c
20    spacing=spacing+step
      do 30 k=1,10
        x(k)=(k-1)*spacing
30    continue
c
      call total_el
c
      if(eold.gt.e_total)then
        eold=e_total
      else
        if(step.gt.-1.0e-3)go to 40
        spacing=spacing-step
        step=step*0.1
      endif
      go to 20
c
40    spacing=spacing+1.0
c
      do 50 k=1,10
        if(d(k).eq.0.0)d(k)=spacing/3.0
50    continue
c
      return
c
      end
```

```
      SUBROUTINE OUTPUT
c
      include 'common.mc'
c
      dum=x(icenter)
      x(icenter)=x(n_points)
      x(n_points)=dum
      dum=y(icenter)
      y(icenter)=y(n_points)
      y(n_points)=dum
      dum=z(icenter)
      z(icenter)=z(n_points)
      z(n_points)=dum
      k=kind(icenter)
      kind(icenter)=kind(n_points)
      kind(n_points)=k
      k=move(icenter)
      move(icenter)=move(n_points)
      move(n_points)=k
      xshift=-x(icenter)
      yshift=-y(icenter)
      zshift=-z(icenter)
c
      write(2,2010)title
2010  format(a)
c
      write(2,2020)ion(1),charge(1),ion(2),charge(2),center_ion,
     1charge(kind(icenter)),n_points,n_per_side,all_charge
2020  format(/'THE POSITIVE ION: ',a30,'CHARGE:',f10.2/
     1'THE NEGATIVE ION: ',a30,'CHARGE:',f10.2/
     2'THE CENTRAL  ION: ',a30,'CHARGE:',f10.2//
     3'THE NUMBER OF IONS:',i10,'  (',i2,' PER SIDE)'//
     4'THE NET CHARGE:',f10.4/)
c
      do 10 k=1,n_points
        x(k)=x(k)+xshift
        y(k)=y(k)+yshift
        z(k)=z(k)+zshift
10    continue
c
      write(2,2030)
2030  format(/'MOVEION:'/)
      ntotal=n_trials*n_points_ml
      write(2,2040)n_trials,maxtrial,ntotal,iseed,move_total,
     1100.0*real(move_total)/real(ntotal),-100*pre_poa,e_moveion,
     2e_moveion*27.2114
2040  format('THE NUMBER OF TRIALS PER ION:',
     1       t36,i10,t50,'( CUT-OFF LIMIT: ',i10,' )'/
     2       'THE TOTAL NUMBER OF TRIALS:',t36,i10,t50,'( SEED: ',
     3       i6,' )'/'THE TOTAL NUMBER OF ACTUAL MOVES:',t36,i10/
     4       'THE PERCENTAGE OF ACTUAL MOVES:',t38,f6.2,' %',t50,
     5       '(PRESET: ',f6.2,' %)'//
     6       'THE TOTAL ENERGY:',f20.4,' HY (',f20.4,' EV )'/)
c
      write(2,2050)
2050  format(/'MOVEION WITH POLARIZATION:'/)
      ntotal=n_trialsp*n_points_ml
      write(2,2040)n_trialsp,maxtrialp,ntotal,iseed,move_totalp,
     1100.0*real(move_totalp)/real(ntotal),-100*pre_poa,e_moveionp,
     2e_moveionp*27.2114
c
      n_trialst=n_trials+n_trialsp
      maxtrialt=maxtrial+maxtrialp
      move_totalt=move_total+move_totalp
```

```
      write(2,2060)
2060  format(/'ALL TOGETHER:'/)
      ntotal=n_trialst*n_points_m1
      write(2,2040)n_trialst,maxtrialt,ntotal,iseed,move_totalt,
     1100.0*real(move_totalt)/real(ntotal),-100*pre_poa,e_moveionp,
     2e_moveionp*27.2114
c
      call count(center_ion,last)
      write(2,2070)center_ion(1:last)
2070  format(/'THE NEAREST NEIGHBORS OF THE CENTRAL ION -- ',a,':'//
     1         t12,'X',t21,'Y',t30,'Z',
     2         t37,'DISTANCE TO CENTER',t60,'AVERAGE',t72,'DEVIATION'/)
c
      weight=1.0
      xdis=x(myfront)-x(icenter)
      ydis=y(myfront)-y(icenter)
      zdis=z(myfront)-z(icenter)
      xtemp(myfront)=sqrt(xdis*xdis+ydis*ydis+zdis*zdis)
      xdis=x(myright)-x(icenter)
      ydis=y(myright)-y(icenter)
      zdis=z(myright)-z(icenter)
      xtemp(myright)=sqrt(xdis*xdis+ydis*ydis+zdis*zdis)
      xdis=x(myabove)-x(icenter)
      ydis=y(myabove)-y(icenter)
      zdis=z(myabove)-z(icenter)
      xtemp(myabove)=sqrt(xdis*xdis+ydis*ydis+zdis*zdis)
      average1=(xtemp(myfront)+xtemp(myright)+xtemp(myabove))/3.0
      deviation=xtemp(myfront)*xtemp(myfront)
     1           +xtemp(myright)*xtemp(myright)
     2           +xtemp(myabove)*xtemp(myabove)
      deviation=deviation/3.0
      deviation=sqrt(deviation-average1*average1)
c
      write(2,2080)x(myfront),y(myfront),z(myfront),xtemp(myfront)
2080  format('FRONT',t8,3(f7.3,2x),t43,f6.3)
      write(2,2090)x(myright),y(myright),z(myright),xtemp(myright),
     1              average1,deviation
2090  format('RIGHT',t8,3(f7.3,2x),t43,f6.3,t60,f6.3,t73,f8.6)
      write(2,2100)x(myabove),y(myabove),z(myabove),xtemp(myabove)
2100  format('ABOVE',t8,3(f7.3,2x),t43,f6.3/)
c
      if(myback.le.0)go to 20
      weight=0.5
      xdis=x(myback)-x(icenter)
      ydis=y(myback)-y(icenter)
      zdis=z(myback)-z(icenter)
      xtemp(myback)=sqrt(xdis*xdis+ydis*ydis+zdis*zdis)
      xdis=x(myleft)-x(icenter)
      ydis=y(myleft)-y(icenter)
      zdis=z(myleft)-z(icenter)
      xtemp(myleft)=sqrt(xdis*xdis+ydis*ydis+zdis*zdis)
      xdis=x(mybelow)-x(icenter)
      ydis=y(mybelow)-y(icenter)
      zdis=z(mybelow)-z(icenter)
      xtemp(mybelow)=sqrt(xdis*xdis+ydis*ydis+zdis*zdis)
      average2=(xtemp(myback)+xtemp(myleft)+xtemp(mybelow))/3.0
      deviation=xtemp(myback)*xtemp(myback)
     1           +xtemp(myleft)*xtemp(myleft)
     2           +xtemp(mybelow)*xtemp(mybelow)
      deviation=deviation/3.0
      deviation=sqrt(deviation-average2*average2)
c
      write(2,2110)x(myback),y(myback),z(myback),xtemp(myback)
2110  format('BACK',t8,3(f7.3,2x),t43,f6.3)
```

```
      write(2,2120)x(myleft),y(myleft),z(myleft),xtemp(myleft),
     1              average2,deviation
2120  format('LEFT',t8,3(f7.3,2x),t43,f6.3,t60,f6.3,t73,f8.6)
      write(2,2130)x(mybelow),y(mybelow),z(mybelow),xtemp(mybelow)
2130  format('BELOW',t8,3(f7.3,2x),t43,f6.3/)
c
20    spacing=(average1+average2)*weight
      write(2,2140)spacing,spacing*0.52917
2140  format('THE AVERAGE NEAREST-NEIGHBOR SPACING:',f9.4,
     1       ' BOHRS (',f9.4,' ANGSTROMS )')
c
      if(ipc.eq.0)return
c
      write(2,2150)
2150  format(//'THE COORDINATES OF THE IONS:'//
     1' ORDER',t11,'ION',t24,'CHARGE',
     2t39,'X',t50,'Y',t61,'Z',t73,'MOVE'/)
c
      do 80 k=1,n_points
        if(kind(k).gt.10)then
          write(2,2160)k,ion(kind(k)-10),charge(kind(k)-10)+charge(5),
     1                 x(k),y(k),z(k),move(k)
2160      format(i6,t11,a6,t22,f8.4,t33,3(f10.4,1x),t71,i6)
        else
          write(2,2160)k,ion(kind(k)),charge(kind(k)),
     1                 x(k),y(k),z(k),move(k)
        endif
80    continue
c
      return
c
      end
```

```
      function ran1(dum)
c
      include 'common.mc'
c
      ix1=mod(ia1*ix1+ic1,m1)
      ix2=mod(ia2*ix2+ic2,m2)
      ix3=mod(ia3*ix3+ic3,m3)
      k=1+97*ix3*rm3
      ran1=random1(k)
      random1(k)=(real(ix1)+real(ix2)*rm2)*rm11
c
      return
c
      end
```

```
      function ran2(dum)
c
      include 'common.mc'
c
      ix1=mod(ia1*ix1+ic1,m1)
      ix2=mod(ia2*ix2+ic2,m2)
      ix3=mod(ia3*ix3+ic3,m3)
      k=1+97*ix3*rm3
      ran2=random2(k)
      random2(k)=(real(ix1)+real(ix2)*rm2)*rm12+negone
c
      return
c
      end
```

```
      SUBROUTINE READIN
c
      include 'common.mc'
c
      character*80 anycha,alphaunit,bunit,cha,cha1,cha2,cunit,dunit,
     1empty,rhounit
c
      parameter(anycha='*',empty=' ')
      parameter(dfactor=1.0/0.529177)
      parameter(dfactor3=dfactor*dfactor*dfactor)
      parameter(dfactor6=dfactor3*dfactor3)
      parameter(efactor=1.0/27.2114)
c
      read(1,1010,end=10)title
1010  format(///a/)
      go to 20
c
10    write(6,6010)
6010  format(///'Sorry. The input file does not exist.'///)
      stop
c
20    read(1,1020)pre_poa,accuracy
1020  format(//'THE PRESET PERCENTAGE OF ACCEPTANCE:',e44.0/
     1          'THE ACCURACY OF THE POSITIONS:',e50.0/)
      if(pre_poa.le.0.0)then
        write(6,6020)pre_poa
6020    format(///'Sorry. The preset percentage of acceptance is:',
     1          t50,f10.2,'%.'//'It must be greater than zero.'//
     2          'Stop.'///)
        stop
      endif
      pre_poa=-0.01*pre_poa
      if(accuracy.lt.0.0)accuracy=-accuracy
      if(accuracy.eq.0.0)accuracy=1.0e-02
c
      read(1,1030)dum,center_ion
1030  format(//'THE NUMBER OF IONS PER SIDE:',e52.0,
     1/'PRINT OUT ALL COORDINATES?',a54/)
      n_per_side=dum
      if(n_per_side.gt.nmax)then
        write(6,6030)nmax,n_per_side
6030    format(///'Sorry. The maximum number of ions per side is',
     1          t50,i3,'.'/'You specified number is',t50,i3,'.'//
     2          'Stop. Increase that number in the file common.mc'/
     3          'and recompile the program.'///)
        stop
      endif
      call cutnull(center_ion)
      if(center_ion(1:1).eq.'Y'.or.center_ion(1:1).eq.'y')ipc=1
c
      if(n_per_side.le.0)then
        write(6,6040)
6040    format(///
     1  'Sorry. The number of points must be greater than zero.'///)
        stop
      endif
c
      read(1,1040)bunit,spacing
1040  format(//'THE INITIAL NEAREST NEIGHBOR SPACING (',t39,a10,') :',
     1        e30.0///)
      if(spacing.lt.0.0)spacing=-spacing
c
      if(spacing.eq.0.0)then
        ionedim=1
```

```
          go to 30
        endif
c
        call cutnull(bunit)
        call casechange(bunit,idum)
        if(bunit(1:1).eq.'A')spacing=spacing*dfactor
c
30      read(1,1050)cha
1050    format(a)
        if(cha(1:7).eq.'LATTICE')then
          alphaunit=cha(61:68)
          call cutnull(alphaunit)
          call casechange(alphaunit,idum)
          go to 30
        endif
        if(cha.ne.' ')then
          n_kinds=n_kinds+1
          call search(cha,1,80,anycha,ilstcha,idum)
          call search(cha,ilstcha,80,empty,iblank,idum)
          ion(n_kinds)=cha(ilstcha:iblank-1)
          call search(cha,iblank,80,anycha,ilstcha,idum)
          call search(cha,ilstcha,80,empty,iblank,idum)
          chal=cha(ilstcha:iblank-1)
          call number(chal,0,dum,idum)
          charge(n_kinds)=dum
          call search(cha,iblank,80,anycha,ilstcha,idum)
          call search(cha,ilstcha,80,empty,iblank,idum)
          chal=cha(ilstcha:iblank-1)
          call number(chal,0,dum,idum)
          alpha(n_kinds)=dum
          call casechange(ion(n_kinds),idum)
          go to 30
        endif
c
        if(charge(1).eq.0.0.or.charge(2).eq.0.0)then
          write(6,6050)
6050      format(///'Sorry. One charge is specified as 0 (zero).'
     1            //'This is not allowed in this program.'///)
          stop
        endif
c
        read(1,1060)center_ion
1060    format(//'THE CENTRAL ION:',a64/)
        call cutnull(center_ion)
        call casechange(center_ion,idum)
        if(center_ion.eq.ion(1))then
          kind_of_center=1
        elseif(center_ion.eq.ion(2))then
          kind_of_center=2
        else
          write(6,6060)
6060      format(///'Sorry. The central ion is not correctly specified.'
     1           ///)
          stop
        endif
c
        read(1,1070)dum
1070    format(//'NUMBER OF DEFECT IONS:',e58.0//)
        n_defects=dum
        if(n_defects.gt.2)then
          write(6,6070)
6070      format(///'Sorry. Too many defect ions.'//
     1           'The maximum number is 2.'///)
          stop
```

```
      endif
      if(n_defects.ne.0)then
        do 40 k=1,n_defects
          n_kinds=n_kinds+1
          read(1,1050)cha
          if(cha.eq.' ')then
            write(6,6090)n_defects
6090        format(///'Sorry. The number of defect ions is',
     1             i2,'.'//
     2                'But there is no entry for the defect ion(s).'
     3              //'Stop.'///)
            stop
          endif
          call search(cha,1,80,anycha,ilstcha,idum)
          call search(cha,ilstcha,80,empty,iblank,idum)
          ion(n_kinds)=cha(ilstcha:iblank-1)
          call casechange(ion(n_kinds),idum)
          call search(cha,iblank,80,anycha,ilstcha,idum)
          call search(cha,ilstcha,80,empty,iblank,idum)
          cha1=cha(ilstcha:iblank-1)
          call number(cha1,0,dum,idum)
          charge(n_kinds)=dum
          call search(cha,iblank,80,anycha,ilstcha,idum)
          call search(cha,ilstcha,80,empty,iblank,idum)
          cha1=cha(ilstcha:iblank-1)
          call number(cha1,0,dum,idum)
          alpha(n_kinds)=dum
          call search(cha,iblank,80,anycha,ilstcha,idum)
          call search(cha,ilstcha,80,empty,iblank,idum)
          defect_site(k)=cha(ilstcha:iblank-1)
          call cutnull(defect_site(k))
          call casechange(defect_site(k),idum)
40      continue
      endif
c
      read(1,1080)bunit,rhounit,cunit,dunit
1080  format(//////'B:',a11,'RHO:',a12,'C:',a23,'D:',a24///)
c
      call cutnull(bunit)
      call cutnull(rhounit)
      call cutnull(cunit)
      call cutnull(dunit)
      call casechange(bunit,idum)
      call casechange(rhounit,idum)
      call casechange(cunit,ic)
      call casechange(dunit,idum)
c
      n_lines=0
c
50    read(1,1050)cha
      if(cha.eq.empty)go to 140
      n_lines=n_lines+1
      call casechange(cha,idum)
      call search(cha,1,80,anycha,ilstcha,idum)
60    call search(cha,ilstcha,80,empty,iblank,idum)
      call search(cha,iblank,80,anycha,ilstcha,idum)
      if(ilstcha.eq.80)go to 50
      do 70 l=1,10
        if(cha(ilstcha:ilstcha).eq.chn(l))go to 80
70    continue
      go to 60
80    n=ilstcha
      if(n.eq.80)then
        write(6,6110)norn(n_lines)
```

```
6110    format(///'Sorry. Error in short-range potentials: ',
     1            a,' line.'//
     2            'Each line should have four numbers.'///)
          stop
        endif
        chal='<>'
        call search(cha,1,iblank,chal,ihead,itail)
        if(ihead.lt.iblank)go to 90
        write(6,6100)norn(n_lines)
6100    format(///'Sorry. Error in short-range potentials: ',a,' line.'/
     1        //'There must be a "<>" between the ion names.'///)
        stop
90      chal=cha(1:ihead-1)
        call cutnull(chal)
        do 100 i=1,n_kinds
          if(chal.eq.ion(i))go to 110
100     continue
        write(6,6120)norn(n_lines),chal
6120    format(///'Sorry. Error in short-range potentials: ',a,' line.'/
     1        //'An ion is not specified:'/a///)
        stop
110     cha2=cha(ihead+2:n-1)
        call cutnull(cha2)
        do 120 j=1,n_kinds
          if(cha2.eq.ion(j))go to 130
120     continue
        write(6,6110)norn(n_lines),cha2
        stop
130     a(ichoose(i,j))=charge(i)*charge(j)
        call search(cha,n,80,empty,iblank,idum)
        chal=cha(n:iblank-1)
        call number(chal,0,dum,idum)
        b(ichoose(i,j))=abs(dum)
        call search(cha,iblank,80,anycha,ilstcha,idum)
        if(ilstcha.eq.80)then
          write(6,6110)norn(n_lines)
          stop
        endif
        call search(cha,ilstcha,80,empty,iblank,idum)
        chal=cha(ilstcha:iblank-1)
        call number(chal,0,dum,idum)
        rho(ichoose(i,j))=abs(dum)
        call search(cha,iblank,80,anycha,ilstcha,idum)
        if(ilstcha.eq.80)then
          write(6,6110)norn(n_lines)
          stop
        endif
        call search(cha,ilstcha,80,empty,iblank,idum)
        chal=cha(ilstcha:iblank-1)
        call number(chal,0,dum,idum)
        c(ichoose(i,j))=abs(dum)
        call search(cha,iblank,80,anycha,ilstcha,idum)
        if(ilstcha.eq.80)then
          write(6,6110)norn(n_lines)
          stop
        endif
        call search(cha,ilstcha,80,empty,iblank,idum)
        chal=cha(ilstcha:iblank-1)
        call number(chal,0,dum,idum)
        d(ichoose(i,j))=abs(dum)
        go to 50
c
140     do 150 k=1,10
          if(a(k).eq.0.0)go to 150
```

```
          if(alphaunit(1:1).eq.'A')alpha(k)=alpha(k)*dfactor3
          if(bunit(1:1).eq.'E')b(k)=b(k)*efactor
          if(bunit(1:1).eq.'R')b(k)=0.5*b(k)
          if(rho(k).eq.0.0)rho(k)=1.0
          if(rhounit(1:1).eq.'A')rho(k)=dfactor*rho(k)
          rho(k)=-1.0/rho(k)
          if(cunit(1:1).eq.'E')c(k)=c(k)*efactor
          if(cunit(ic:ic).eq.'A')c(k)=c(k)*dfactor6
          if(d(k).eq.0.0)d(k)=spacing/3.0
          if(dunit(1:1).eq.'A')d(k)=d(k)*dfactor
150   continue
c
      return
c
      end
```

```
      SUBROUTINE SEARCH(CHA,ISTART,IFINISH,SCHA,IHEAD,ITAIL)
c
      character*80 cha,scha,temp
c
      ilst=0
      last=0
c
      do 10 k=1,80
        if(scha(k:k).ne.' ')then
          ilst=k
          go to 20
        endif
10    continue
      ilst=80
c
20    do 30 k=ilst,80
        if(scha(k:k).eq.' ')then
          last=k-1
          go to 40
        endif
30    continue
      last=80
c
40    temp=scha(ilst:last)
      numcha=last-ilst+1
      if(temp.eq.' ')numcha=0
c
      do 50 k=istart,ifinish-numcha
        if(temp.eq.'*')then
          if(cha(k:k).ne.' ')then
            ihead=k
            itail=k
            return
          endif
          go to 50
        endif
        kpnumcha=k+numcha
        if(cha(k:kpnumcha).eq.temp)then
          ihead=k
          itail=kpnumcha
          return
        endif
50    continue
c
      ihead=ifinish
      itail=ifinish
c
      return
c
      end
```

```
      SUBROUTINE SHUFFLE
c
      include 'common.mc'
c
      do 10 k=1,n_points_m1
        xtemp(k)=ran1(dum)
10    continue
c
      do 30 i=1,n_points_m2
        do 20 j=i+1,n_points_m1
          if(xtemp(j).gt.xtemp(i))go to 20
          dum=xtemp(i)
          xtemp(i)=xtemp(j)
          xtemp(j)=dum
          k=iorder(i)
          iorder(i)=iorder(j)
          iorder(j)=k
20      continue
30    continue
c
      return
c
      end
```

```
      SUBROUTINE TOTAL_E
c
      include 'common.mc'
c
      e_total=0.0
c
      do 20 ii=1,n_points
        do 10 jj=ii+1,n_points
          dx=x(ii)-x(jj)
          dy=y(ii)-y(jj)
          dz=z(ii)-z(jj)
          sr2=dx*dx+dy*dy+dz*dz
          if(sr2.lt.1.0e+2)then
            kk=sr2*10000+0.5
            sr=sroot(kk)
          elseif(sr2.lt.1.0e+4)then
            kk=sr2*100+0.5
            sr=10.0*sroot(kk)
          elseif(sr2.lt.1.0e+6)then
            kk=sr2+0.5
            sr=100.0*sroot(kk)
          elseif(sr2.lt.1.0e+8)then
            kk=sr2*1.0e-2+0.5
            sr=1000.0*sroot(kk)
          else
            write(2,2010)ii,jj,x(ii),x(jj),dx,y(ii),y(jj),dy,
     1                 z(ii),z(jj),dz,sr2
2010        format(///'Sorry. SROOT is not large enough for TOTAL_E.'//
     1             'i=',i4,' j=',i4/
     2             'x(i)=',f20.8,' x(j)=',f20.8,' dx=',f20.8/
     3             'y(i)=',f20.8,' y(j)=',f20.8,' dy=',f20.8/
     4             'z(i)=',f20.8,' z(j)=',f20.8,' dz=',f20.8/
     5             'sr2=',f20.8)
            stop
          endif
          if(sr.gt.100.0)then
            e_total=e_total
     1             +a(ichoose(kind(ii),kind(jj)))/sr
          else
            kk=sr*10000
            e_total=e_total
     1             +potential(kk,ichoose(kind(ii),kind(jj)))
          endif
10      continue
20    continue
c
      return
c
      end
```

```
      SUBROUTINE TOTAL_E1
c
      include 'common.mc'
c
      e_total=0.0
c
      do 30 ii=1,10
        do 20 jj=ii+1,10
          c5=0.0
          if(kind(ii).gt.10)then
            kind_ii=kind(ii)-10
            if(kind(jj).gt.10)then
              kind_jj=kind(jj)-10
              c5=a(ichoose(5,kind_ii))+a(ichoose(5,kind_jj))+a(15)
            else
              kind_jj=kind(jj)
              c5=a(ichoose(5,kind_jj))
            endif
            go to 10
          else
            kind_ii=kind(ii)
          endif
          if(kind(jj).gt.10)then
            kind_jj=kind(jj)-10
            c5=a(ichoose(5,kind_ii))
          else
            kind_jj=kind(jj)
          endif
10        dx=x(ii)-x(jj)
          distance=sqrt(dx*dx)
          if(distance.gt.100.0)then
            e_total=e_total+a(ichoose(kind_ii,kind_jj))/distance
          else
            kk=distance*10000
            e_total=e_total+potential(kk,ichoose(kind_ii,kind_jj))
          endif
          if(c5.ne.0.0)e_total=e_total+c5/distance
20      continue
30    continue
c
      return
c
      end
```

```
      SUBROUTINE TOTAL_EC
c
      include 'common.mc'
c
      e_total=0.0
c
      do 30 ii=1,n_points
        do 20 jj=ii+1,n_points
          c5=0.0
          if(kind(ii).gt.10)then
            kind_ii=kind(ii)-10
            if(kind(jj).gt.10)then
              kind_jj=kind(jj)-10
              c5=a(ichoose(5,kind_ii))+a(ichoose(5,kind_jj))+a(15)
            else
              kind_jj=kind(jj)
              c5=a(ichoose(5,kind_jj))
            endif
            go to 10
          else
            kind_ii=kind(ii)
          endif
          if(kind(jj).gt.10)then
            kind_jj=kind(jj)-10
            c5=a(ichoose(5,kind_ii))
          else
            kind_jj=kind(jj)
          endif
10        dx=x(ii)-x(jj)
          dy=y(ii)-y(jj)
          dz=z(ii)-z(jj)
          sr2=dx*dx+dy*dy+dz*dz
          if(sr2.lt.1.0e+2)then
            kk=sr2*10000+0.5
            sr=sroot(kk)
          elseif(sr2.lt.1.0e+4)then
            kk=sr2*100+0.5
            sr=10.0*sroot(kk)
          elseif(sr2.lt.1.0e+6)then
            kk=sr2+0.5
            sr=100.0*sroot(kk)
          elseif(sr2.lt.1.0e+8)then
            kk=sr2*1.0e-2+0.5
            sr=1000.0*sroot(kk)
          else
            write(2,2010)ii,jj,x(ii),x(jj),dx,y(ii),y(jj),dy,
     1                   z(ii),z(jj),dz,sr2
2010        format(///'Sorry. SROOT is not large enough for TOTAL_EC.'//
     1               'i=',i4,' j=',i4/
     2               'x(i)=',f20.8,' x(j)=',f20.8,' dx=',f20.8/
     3               'y(i)=',f20.8,' y(j)=',f20.8,' dy=',f20.8/
     4               'z(i)=',f20.8,' z(j)=',f20.8,' dz=',f20.8/
     5               'sr2=',f20.8)
            stop
          endif
          if(sr.gt.100.0)then
            e_total=e_total
     1             +a(ichoose(kind_ii,kind_jj))/sr
          else
            kk=sr*10000
            e_total=e_total
     1             +potential(kk,ichoose(kind_ii,kind_jj))
          endif
          if(c5.ne.0.0)e_total=e_total+c5/sr
```

```
20        continue
30     continue
c
       return
c
       end
```

```
      SUBROUTINE TOTAL_EP
c
      include 'common.mc'
c
      call getp
c
      e_total=0.0
c
      do 20 ii=1,n_points
        do 10 jj=ii+1,n_points
          dx=x(ii)-x(jj)
          dy=y(ii)-y(jj)
          dz=z(ii)-z(jj)
          sr2=dx*dx+dy*dy+dz*dz
          if(sr2.lt.1.0e+2)then
            kk=sr2*10000+0.5
            sr=sroot(kk)
          elseif(sr2.lt.1.0e+4)then
            kk=sr2*100+0.5
            sr=10.0*sroot(kk)
          elseif(sr2.lt.1.0e+6)then
            kk=sr2+0.5
            sr=100.0*sroot(kk)
          elseif(sr2.lt.1.0e+8)then
            kk=sr2*1.0e-2+0.5
            sr=1000.0*sroot(kk)
          else
            write(2,2010)ii,jj,x(ii),x(jj),dx,y(ii),y(jj),dy,
     1                   z(ii),z(jj),dz,sr2
2010        format(///'Sorry. SROOT is not large enough for TOTAL_EP.'//
     1               'i=',i4,' j=',i4/
     2               'x(i)=',f20.8,' x(j)=',f20.8,' dx=',f20.8/
     3               'y(i)=',f20.8,' y(j)=',f20.8,' dy=',f20.8/
     4               'z(i)=',f20.8,' z(j)=',f20.8,' dz=',f20.8/
     5               'sr2=',f20.8)
            stop
          endif
          sr3=sr*sr2
          uvx=dx/sr
          uvy=dy/sr
          uvz=dz/sr
          pi_dot_pj=p(1,ii)*p(1,jj)+p(2,ii)*p(2,jj)+p(3,ii)*p(3,jj)
          pi_dot_rij=p(1,ii)*uvx+p(2,ii)*uvy+p(3,ii)*uvz
          pj_dot_rij=p(1,jj)*uvx+p(2,jj)*uvy+p(3,jj)*uvz
          w=(pi_dot_pj-3.0*pi_dot_rij*pj_dot_rij)/sr3
          if(sr.gt.100.0)then
            e_total=e_total
     1              +a(ichoose(kind(ii),kind(jj)))/sr+w
          else
            kk=sr*10000
            e_total=e_total
     1              +potential(kk,ichoose(kind(ii),kind(jj)))+w
          endif
10      continue
20    continue
c
      return
c
      end
```

```
      SUBROUTINE TOTAL_EPC
c
      include 'common.mc'
c
      call getp
c
      e_total=0.0
c
      do 30 ii=1,n_points
        do 20 jj=ii+1,n_points
          c5=0.0
          if(kind(ii).gt.10)then
            kind_ii=kind(ii)-10
            if(kind(jj).gt.10)then
              kind_jj=kind(jj)-10
              c5=a(ichoose(5,kind_ii))+a(ichoose(5,kind_jj))+a(15)
            else
              kind_jj=kind(jj)
              c5=a(ichoose(5,kind_jj))
            endif
            go to 10
          else
            kind_ii=kind(ii)
          endif
          if(kind(jj).gt.10)then
            kind_jj=kind(jj)-10
            c5=a(ichoose(5,kind_ii))
          else
            kind_jj=kind(jj)
          endif
10        dx=x(ii)-x(jj)
          dy=y(ii)-y(jj)
          dz=z(ii)-z(jj)
          sr2=dx*dx+dy*dy+dz*dz
          if(sr2.lt.1.0e+2)then
            kk=sr2*10000+0.5
            sr=sroot(kk)
          elseif(sr2.lt.1.0e+4)then
            kk=sr2*100+0.5
            sr=10.0*sroot(kk)
          elseif(sr2.lt.1.0e+6)then
            kk=sr2+0.5
            sr=100.0*sroot(kk)
          elseif(sr2.lt.1.0e+8)then
            kk=sr2*1.0e-2+0.5
            sr=1000.0*sroot(kk)
          else
            write(2,2010)ii,jj,x(ii),x(jj),dx,y(ii),y(jj),dy,
     1                   z(ii),z(jj),dz,sr2
2010        format(///
     1             'Sorry. SROOT is not large enough for TOTAL_EPC.'//
     2             'i=',i4,' j=',i4/
     3             'x(i)=',f20.8,' x(j)=',f20.8,' dx=',f20.8/
     4             'y(i)=',f20.8,' y(j)=',f20.8,' dy=',f20.8/
     5             'z(i)=',f20.8,' z(j)=',f20.8,' dz=',f20.8/
     6             'sr2=',f20.8)
            stop
          endif
          sr3=sr*sr2
          uvx=dx/sr
          uvy=dy/sr
          uvz=dz/sr
          pi_dot_pj=p(1,ii)*p(1,jj)+p(2,ii)*p(2,jj)+p(3,ii)*p(3,jj)
          pi_dot_rij=p(1,ii)*uvx+p(2,ii)*uvy+p(3,ii)*uvz
```

```
          pj_dot_rij=p(1,jj)*uvx+p(2,jj)*uvy+p(3,jj)*uvz
          w=(pi_dot_pj-3.0*pi_dot_rij*pj_dot_rij)/sr3
          if(sr.gt.100.0)then
            e_total=e_total
     1            +a(ichoose(kind_ii,kind_jj))/sr+w
          else
            kk=sr*10000
            e_total=e_total
     1            +potential(kk,ichoose(kind_ii,kind_jj))+w
          endif
          if(c5.ne.0.0)e_total=e_total+c5/sr
20      continue
30    continue
c
      return
c
      end
```

APPENDIX B
LISTING OF THE MAIN PROGRAM LOPAS CODE
UNI-PROCESSOR VERSION

A. B. Kunz, Author

deserver:barry

lopas

Fri Mar 8 13:08:47 1991

lw / TCD LaserWriter II NT

```
lw deserver:barry  Job: lopas  Date: Fri Mar  8 13:08:47 1991

lw deserver:barry  Job: lopas  Date: Fri Mar  8 13:08:47 1991

lw deserver:barry  Job: lopas  Date: Fri Mar  8 13:08:47 1991

lw deserver:barry  Job: lopas  Date: Fri Mar  8 13:08:47 1991
```

```
c
c     This is an implementation
c     of local orbitals procedures of
c     Adams, Gilbert and Kunz implemented
c     for cluster building blocks and
c     a gaussian basis set
c     part of the MEGAMOL sequence
c     Molecules for the 90's
c     author is A B Kunz
c     Michigan Technological University
c     College of Engineering
c     Fortran 77
c     written 1989-91
c     all rights reserved by the author
c
c     *****************************************************************
c
      Program lopas
c
c     *****************************************************************
c
      implicit real*8(a-h,o-z)
      dimension nenv(20),id(20,200),
     1xof(20,200),yof(20,200),zof(20,200),
     2a(20,200),b(20,200),c(20,200)
      common/angle/angl(73),cangl(73),sangl(73)
      real*8 norm
c
c     *****************************************************************
c
    1 format(i4)
    2 format(' THIS IS A GAUSSIAN BASIS SET LOPAS CALCULATION ',/,
     1' USING THE MULTI CENTER METHOD OF A B KUNZ ',/,
     2' FOLLOWING THE PROCEDURE OF ADAMS-GILBERT-KUNZ ')
    3 format(1x,' nbb = ',i4)
    4 format(i4)
    5 format(i4,6x,6f10.4)
    6 format(1x,' nenv(i) = ',i4)
    7 format(1x,' id = ',i4,' xof = ',f10.4,' yof = ',f10.4,
     1' zof = ',f10.4,/,'  angle 1 = ',f10.4,' angle 2 = ',f10.4,
     2' angle 3 = ',f10.4)
   19 format (' moments run ')
   18 format (' pot run ')
c
c     *****************************************************************
c
      open(unit=61,file='mol5f.dat',form='formatted')
      open(unit=60,file='mol5e.dat',form='formatted')
      open(unit=14,file='mol14.dat',form='formatted')
c
c     *****************************************************************
c
c     define local orbitals problem
c
      read(14,1)nbb
      write(60,2)
      write(60,3)nbb
      print 2
      print 3,nbb
      if(nbb.lt.1.or.nbb.gt.20)stop ' wrong number of building blocks '
      do 20 i=1,nbb
      read(14,4)nenv(i)
      if (nenv(i).gt.200)stop ' too many in environment '
      do 20 j=1,nenv (i)
      read(14,5)id(i,j),xof(i,j),yof(i,j),zof(i,j),a(i,j),b(i,j),c(i,j)
   20 continue
```

```
      do 21 i=1,nbb
      write(60,6)nenv(i)
      print 6,nenv(i)
      do 21 j=1,nenv(i)
      write(60,7)id(i,j),xof(i,j),yof(i,j),zof(i,j),a(i,j),b(i,j),c(i,j)
   21 print 7,id(i,j),xof(i,j),yof(i,j),zof(i,j),a(i,j),b(i,j),c(i,j)
      close(unit=60)
      close(unit=14)
C
C*******************************************************************
C
C     set up angular table here
C
      do 101 i=1,73
      angl(i)=float(i-1)*0.087266463
      cangl(i)=cos(angl(i))
  101 sangl(i)=sin(angl(i))
C
C     lopas set up data done now
C     Do local orbital building blocks in free space now
C
C     ************************************************************
C
      ilop=0
      do 30 i=1,nbb
C*****this is a par-do
      iread=0
      call lister(i,ilop)
      call poly(i,ilop,iread,nbfs,non)
      call uhf(i,ilop)
   30 continue
C     free space estimates of building blocks are
C     evaluated here
C     get multipole moments and begin lopas rotations
C
C     ************************************************************
C
      do 9999 ilps=1,4
      ilop=ilps
C     evaluate moments of each lopas block here
C     evaluate detailed potentials as well
      do 40 i=1,nbb
C*****this is a par-do
   40 call moments(nbfs,i,ilps)
      print 19
C     moments and v00 potential formed for each block
C     form potential of environment of each building block
      do 50 i=1,nbb
C*****this is a par-do
      call pot(nbfs,ilps,nenv,xof,yof,zof,a,b,c,i,id)
      print 18
      call uhf(i,ilop)
   50 continue
 9999 continue
      close(unit=61)
      stop 'lopas complete'
      end
      Subroutine moments(nbfs,ibb,ilpss)
c     calculates V00 potential for this building block
c     calculates moments as well
c     uses spherical coordinates and 3-d numerical quadrature
c     authored by A B Kunz
c     Fortran 77
c     all rights reserved by the author
c     integrations use Weddle's rule over angles
c     integrations use Simpson's rule over r
```

APPENDIX C

LISTING OF THE LOPAS PROGRAM SUBROUTINES

FOR BOTH

UNI-PROCESSOR and PARALLEL PROCESSOR

A. B. Kunz, Author

```
      Subroutine moments(nbfs,ibb,ilpss)
c     calculates V00 potential for this building block
c     calculates moments as well
c     uses spherical coordinates and 3-d numerical quadrature
c     authored by A B Kunz
c     Fortran 77
c     all rights reserved by the author
c     integrations use Weddle's rule over angles
c     integrations use Simpson's rule over r
c     weighting factors are used to define each
c     specific integrations properties and as an
c     aid to high speed computation. This reduces each
c     integral to a dot product, and facilitates vectorization
c     by a smart compiler
c
c***********************************************************************
c
      implicit real*8(a-h,o-z)
      dimension rho(81,37,73),r(81),ri1(37,73),ri2(73),wr(81),
     1wa1(37),wa2(73),ntype(180),nfirst(180),nlast(180),
     2fodm(180,180),t1(218781),t2(218781),t3(218781),nr(20
     3,3),t5(81,37,73),t4(81,37,73),eta(1024,5),c(1024)
     4,elmom(4,6,6,6),ps(360),v(81),v1(81),v2(81)
      common/angle/angl(73),cangl(73),sangl(73)
      common/mompot/vlist(1024,4),ntype,nfirst,nlast,eta,c
      real*8 norm
      real*4 ain(180),psi(2,180,180)
      character*4 zl(20)
      character*15 mol51(20)
      character*11 mol5a(20),mol11(20),mol04(20),mol30(20),mol41(20),
     1mol40(20)
      equivalence(t3(1),rho(1,1,1))
      equivalence(t4(1,1,1),t1(1))
      equivalence(t5(1,1,1),t2(1))
      data  nr /  0,1,0,0,2,0,0 1,1,0,3,0,0,2,2,1,0,1,0,1,
     x            0,0,1,0,0,2,0,1,0,1,0,3,0,1,0,2,2,0,1,1,
     x            0,0,0,1,0,0,2,0,1,1,0,0,3,0,1,0,1,2,2,1 /
c
c***********************************************************************
c
      mol51(1)='/work/psi01.dat'
      mol51(2)='/work/psi02.dat'
      mol51(3)='/work/psi03.dat'
      mol51(4)='/work/psi04.dat'
      mol51(5)='/work/psi05.dat'
      mol51(6)='/work/psi06.dat'
      mol51(7)='/work/psi07.dat'
      mol51(8)='/work/psi08.dat'
      mol51(9)='/work/psi09.dat'
      mol51(10)='/work/psi10.dat'
      mol51(11)='/work/psi11.dat'
      mol51(12)='/work/psi12.dat'
      mol51(13)='/work/psi13.dat'
      mol51(14)='/work/psi14.dat'
      mol51(15)='/work/psi15.dat'
      mol51(16)='/work/psi16.dat'
      mol51(17)='/work/psi17.dat'
      mol51(18)='/work/psi18.dat'
      mol51(19)='/work/psi19.dat'
      mol51(20)='/work/psi20.dat'
      mol5a(1)='mol5a01.dat'
      mol5a(2)='mol5a02.dat'
      mol5a(3)='mol5a03.dat'
      mol5a(4)='mol5a04.dat'
```

```
mol5a(5)='mol5a05.dat'
mol5a(6)='mol5a06.dat'
mol5a(7)='mol5a07.dat'
mol5a(8)='mol5a08.dat'
mol5a(9)='mol5a09.dat'
mol5a(10)='mol5a10.dat'
mol5a(11)='mol5a11.dat'
mol5a(12)='mol5a12.dat'
mol5a(13)='mol5a13.dat'
mol5a(14)='mol5a14.dat'
mol5a(15)='mol5a15.dat'
mol5a(16)='mol5a16.dat'
mol5a(17)='mol5a17.dat'
mol5a(18)='mol5a18.dat'
mol5a(19)='mol5a19.dat'
mol5a(20)='mol5a20.dat'
mol11(1)='mol1101.dat'
mol11(2)='mol1102.dat'
mol11(3)='mol1103.dat'
mol11(4)='mol1104.dat'
mol11(5)='mol1105.dat'
mol11(6)='mol1106.dat'
mol11(7)='mol1107.dat'
mol11(8)='mol1108.dat'
mol11(9)='mol1109.dat'
mol11(10)='mol1110.dat'
mol11(11)='mol1111.dat'
mol11(12)='mol1112.dat'
mol11(13)='mol1113.dat'
mol11(14)='mol1114.dat'
mol11(15)='mol1115.dat'
mol11(16)='mol1116.dat'
mol11(17)='mol1117.dat'
mol11(18)='mol1118.dat'
mol11(19)='mol1119.dat'
mol11(20)='mol1120.dat'
mol30(1)='mol3001.dat'
mol30(2)='mol3002.dat'
mol30(3)='mol3003.dat'
mol30(4)='mol3004.dat'
mol30(5)='mol3005.dat'
mol30(6)='mol3006.dat'
mol30(7)='mol3007.dat'
mol30(8)='mol3008.dat'
mol30(9)='mol3009.dat'
mol30(10)='mol3010.dat'
mol30(11)='mol3011.dat'
mol30(12)='mol3012.dat'
mol30(13)='mol3013.dat'
mol30(14)='mol3014.dat'
mol30(15)='mol3015.dat'
mol30(16)='mol3016.dat'
mol30(17)='mol3017.dat'
mol30(18)='mol3018.dat'
mol30(19)='mol3019.dat'
mol30(20)='mol3020.dat'
mol04(1)='mol0401.dat'
mol04(2)='mol0402.dat'
mol04(3)='mol0403.dat'
mol04(4)='mol0404.dat'
mol04(5)='mol0405.dat'
mol04(6)='mol0406.dat'
mol04(7)='mol0407.dat'
mol04(8)='mol0408.dat'
```

```
      mol04(9)='mol0409.dat'
      mol04(10)='mol0410.dat'
      mol04(11)='mol0411.dat'
      mol04(12)='mol0412.dat'
      mol04(13)='mol0413.dat'
      mol04(14)='mol0414.dat'
      mol04(15)='mol0415.dat'
      mol04(16)='mol0416.dat'
      mol04(17)='mol0417.dat'
      mol04(18)='mol0418.dat'
      mol04(19)='mol0419.dat'
      mol04(20)='mol0420.dat'
      mol40(1)='mol4001.dat'
      mol40(2)='mol4002.dat'
      mol40(3)='mol4003.dat'
      mol40(4)='mol4004.dat'
      mol40(5)='mol4005.dat'
      mol40(6)='mol4006.dat'
      mol40(7)='mol4007.dat'
      mol40(8)='mol4008.dat'
      mol40(9)='mol4009.dat'
      mol40(10)='mol4010.dat'
      mol40(11)='mol4011.dat'
      mol40(12)='mol4012.dat'
      mol40(13)='mol4013.dat'
      mol40(14)='mol4014.dat'
      mol40(15)='mol4015.dat'
      mol40(16)='mol4016.dat'
      mol40(17)='mol4017.dat'
      mol40(18)='mol4018.dat'
      mol40(19)='mol4019.dat'
      mol40(20)='mol4020.dat'
      mol41(1)='mol4101.dat'
      mol41(2)='mol4102.dat'
      mol41(3)='mol4103.dat'
      mol41(4)='mol4104.dat'
      mol41(5)='mol4105.dat'
      mol41(6)='mol4106.dat'
      mol41(7)='mol4107.dat'
      mol41(8)='mol4108.dat'
      mol41(9)='mol4109.dat'
      mol41(10)='mol4110.dat'
      mol41(11)='mol4111.dat'
      mol41(12)='mol4112.dat'
      mol41(13)='mol4113.dat'
      mol41(14)='mol4114.dat'
      mol41(15)='mol4115.dat'
      mol41(16)='mol4116.dat'
      mol41(17)='mol4117.dat'
      mol41(18)='mol4118.dat'
      mol41(19)='mol4119.dat'
      mol41(20)='mol4120.dat'
c
c***********************************************************************
c
c     set up initial integration meshes now
c     linear meshes over angle
c     logarithmic mesh over radius
      h=-6.9
      del=0.12375
      r(1)=exp(h)
      do 100 i=3,81,2
      h=h+del+del
      r(i)=exp(h)
```

```
 100  r(i-1)=0.5*(r(i)+r(i-2))
      dang=0.087266463
c     set up integration weight tables
c     angular integrals use weddle's rule on equal integrvals
c     and theta integral is sin(theta) weighted as well
c     radial integrals are using simpson's rule for changing mesh
      wr(1)=(r(2)-r(1))/3.0
      wr(81)=(r(81)-r(80))/3.0
      do 101 i=2,78,2
      wr(i)=(r(i)-r(i-1))*1.3333333333333
 101  wr(i+1)=(r(i+2)-r(i))/3.0
      wr(80)=(r(81)-r(80))*1.333333333333
      do 99 i=1,81
  99  wr(i)=wr(i)*r(i)*r(i)
      do 102 i=1,36,6
      wa1(i)=2.0
      wa1(i+1)=5.0
      wa1(i+2)=1.0
      wa1(i+3)=6.0
      wa1(i+4)=1.0
      wa1(i+5)=5.0
 102  continue
      wa1(1)=1.0
      wa1(37)=1.0
      dmul=0.3*dang
      do 103 i=1,37
      th=float(i-1)*dang
 103  wa1(i)=wa1(i)*dmul*sin(th)
      do 104 i=1,72,6
      wa2(i)=2.0
      wa2(i+1)=5.0
      wa2(i+2)=1.0
      wa2(i+3)=6.0
      wa2(i+4)=1.0
      wa2(i+5)=5.0
 104  continue
      wa2(1)=1.0
      wa2(73)=1.0
      do 105 i=1,73
 105  wa2(i)=dmul*wa2(i)
c     integration factors set
      nfmx=180
      ngmx=1024
      ncmx=1024
      ntmx=20
      maxtyp=0
      ilopas=0
      iread=1
      call poly(ibb,ilopas,iread,nbfs,non)
      close(unit=60)
      close (unit=5)
c
c***********************************************************************
c
c     form electronic charge density now
c     read in wave function coeficients
c
c***********************************************************************
c
      open(unit=4,file=mol04(ibb),form='formatted')
      read(4,108)zl
      read(4,109)nup,ndn,nbas
      k=0
  70  format(8i4)
```

```
      close(unit=4)
      open(unit=30,file=mol30(ibb),form='unformatted',
     1access='direct',recl=720)
      id30=3
      do 106 i=1,nup
      read(30,rec=id30)ain
      id30=id30+1
      do 106 j=1,nbas
 106  psi(1,i,j)=ain(j)
      id30=nbas+3
      do 107 i=1,ndn
      read(30,rec=id30)ain
      id30=id30+1
      do 107 j=1,nbas
 107  psi(2,i,j)=ain(j)
      close(unit=30)
c
c***********************************************************************
c
c     all coeficients available
c     zero rho here
c
c***********************************************************************
c
      do 110 i=1,81
      do 110 j=1,37
      do 110 k=1,73
 110  rho(i,j,k)=0.0
 108  format(20a4)
 109  format(20i4)
      open(unit=51,file=mol51(ibb),form='unformatted',
     1access='direct',recl=1750248)
c
c     form basis functions first
c
      if(ilpss.gt.1) go to 750
      do 1100 nn=1,nbfs
      il=ntype(nn)
      l1=nr(il,1)
      m1=nr(il,2)
      n1=nr(il,3)
      do 1110 i=1,81
      rr=r(i)
      do 1110 j=1,37
      xy=rr*sangl(j)
      zz=rr*cangl(j)
      do 1110 k=1,73
      t4(i,j,k)=0.0
      xx=xy*cangl(k)
      yy=xy*sangl(k)
c     cartesian coordinates are formed
c     get amplitude of basis functions here
      do 1110 l=nfirst(nn),nlast(nn)
      x=eta(l,1)
      y=eta(l,2)
      z=eta(l,3)
      expnt=eta(l,4)
      xnorm=eta(l,5)
      dx=xx-x
      dy=yy-y
      dz=zz-z
      dr=dx*dx+dy*dy+dz*dz
      go to(3000,3001,3002,3003,3004,3005,3006,3007,3008,3009,3010,
     13011,3012,3013,3014,3015,3016,3017,3018,3019)il
```

```
 3000 angr=xnorm
      go to 3020
 3001 angr=xnorm*dx
      go to 3020
 3002 angr=xnorm*dy
      go to 3020
 3003 angr=xnorm*dz
      go to 3020
 3004 angr=xnorm*dx*dx
      go to 3020
 3005 angr=xnorm*dy*dy
      go to 3020
 3006 angr=xnorm*dz*dz
      go to 3020
 3007 angr=xnorm*dx*dy
      go to 3020
 3008 angr=xnorm*dx*dz
      go to 3020
 3009 angr=xnorm*dy*dz
      go to 3020
 3010 angr=xnorm*dx*dx*dx
      go to 3020
 3011 angr=xnorm*dy*dy*dy
      go to 3020
 3012 angr=xnorm*dz*dz*dz
      go to 3020
 3013 angr=xnorm*dx*dx*dy
      go to 3020
 3014 angr=xnorm*dx*dx*dz
      go to 3020
 3015 angr=xnorm*dx*dy*dy
      go to 3020
 3016 angr=xnorm*dy*dy*dz
      go to 3020
 3017 angr=xnorm*dx*dz*dz
      go to 3020
 3018 angr=xnorm*dy*dz*dz
      go to 3020
 3019 angr=xnorm*dx*dy*dz
 3020 continue
      rad=exp(-expnt*dr)
 1110 t4(i,j,k)=t4(i,j,k)+angr*rad
c
c     wave function formed, write it to disc
c
 1100 write(51,rec=nn)t1
  750 continue
c
c     form electronic charge density now
c
      do 121 i=1,218781
  121 t3(i)=0.0
      do 122 i=1,nup
      do 123 j=1,218781
  123 t2(j)=0.0
      do 124 j=1,nbas
      read(51,rec=j)t1
      do 124 k=1,218781
  124 t2(k)=t2(k)+t1(k)*psi(1,i,j)
      do 122 k=1,218781
  122 t3(k)=t3(k)+t2(k)*t2(k)
      do 125 i=1,ndn
      do 126 j=1,218781
  126 t2(j)=0.0
```

```
      do 127 j=1,nbas
      read(51,rec=j)t1
      do 127 k=1,218781
  127 t2(k)=t2(k)+t1(k)*psi(2,i,j)
      do 125 k=1,218781
  125 t3(k)=t3(k)+t2(k)*t2(k)
      close(unit=51)
c
c*********************************************************************
c
c     electronic charge density formed
c     compute moments
c     compute V00 potential
c
c*********************************************************************
c
c     get the monopole moment
      do 131 k=1,73
      do 131 j=1,37
      ans=0.0
      do 132 i=1,81
 132  ans=ans+rho(i,j,k)*wr(i)
 131  ri1(j,k)=ans
      do 133 k=1,73
      ans=0.0
      do 134 j=1,37
 134  ans=ans+ri1(j,k)*wa1(j)
 133  ri2(k)=ans
      ans=0.0
      do 135 k=1,73
 135  ans=ans+ri2(k)*wa2(k)
c     electronic part of monopole moment complete
c     add in the nuclear part
      sum=0.0
      do 136 i=1,non
 136  sum=sum+vlist(i,4)
c     nuclear part on hand
c     monopole moment is xmon=sum-ans
      xmon=sum-ans
c     monopole moment found
c     get dipole moments next
c     do electronic part first
      elx=0.0
      ely=0.0
      elz=0.0
      open(unit=40,file=mol40(ibb),form='unformatted')
      write(40)rho
c     px first
      do 140 i=1,81
      do 140 j=1,37
      th=float(j-1)*dang
      sth=sangl(j)
      do 140 k=1,73
      phi=float(k-1)*dang
 140  rho(i,j,k)=rho(i,j,k)*cangl(k)*sth*r(i)
      do 141 k=1,73
      do 141 j=1,37
      ans=0.0
      do 142 i=1,81
 142  ans=ans+rho(i,j,k)*wr(i)
 141  ri1(j,k)=ans
      do 143 k=1,73
      ans=0.0
      do 144 j=1,37
```

```
 144  ans=ans+ri1(j,k)*wa1(j)
 143  ri2(k)=ans
      do 145 k=1,73
 145  elx=elx+ri2(k)*wa2(k)
c     do py next
      rewind 40
      read(40)rho
      do 150 i=1,81
      do 150 j=1,37
      th=float(j-1)*dang
      sth=sangl(j)
      do 150 k=1,73
      phi=float(k-1)*dang
 150  rho(i,j,k)=rho(i,j,k)*sangl(k)*sth*r(i)
      do 151 k=1,73
      do 151 j=1,37
      ans=0.0
      do 152 i=1,81
 152  ans=ans+rho(i,j,k)*wr(i)
 151  ri1(j,k)=ans
      do 153 k=1,73
      ans=0.0
      do 154 j=1,37
 154  ans=ans+ri1(j,k)*wa1(j)
 153  ri2(k)=ans
      do 155 k=1,73
 155  ely=ely+ri2(k)*wa2(k)
c     do pz next
      rewind 40
      read(40)rho
      do 160 i=1,81
      do 160 j=1,37
      th=float(j-1)*dang
      cth=cangl(j)
      do 160 k=1,73
 160  rho(i,j,k)=rho(i,j,k)*cth*r(i)
      do 161 k=1,73
      do 161 j=1,37
      ans=0.0
      do 162 i=1,81
 162  ans=ans+rho(i,j,k)*wr(i)
 161  ri1(j,k)=ans
      do 163 k=1,73
      ans=0.0
      do 164 j=1,37
 164  ans=ans+ri1(j,k)*wa1(j)
 163  ri2(k)=ans
      do 165 k=1,73
 165  elz=elz+ri2(k)*wa2(k)
c     add in nuclear part next
c     px,py,pz together
      xneg=-1.0
      elx=elx*xneg
      ely=ely*xneg
      elz=elz*xneg
      do 170 i=1,non
      elx=elx+vlist(i,4)*vlist(i,1)
      ely=ely+vlist(i,4)*vlist(i,2)
 170  elz=elz+vlist(i,4)*vlist(i,3)
      open(unit=60,file='moments.dat',form='formatted')
      write(60,171)xmon,elx,ely,elz
      print 171,xmon,elx,ely,elz
 171  format(1x,' net charge = ',f12.4,//,
     1'  px = ',f12.4,'  py = ',f12.4,'  pz = ',f12.4)
```

```
c
c     do the general multipole term now
c
      lmax=5
      if (lmax.le.1) go to 207
          do 200 m=2,lmax
          do 200 n1=0,m
          do 200 n2=0,m
          do 200 n3=0,m
          if((n1+n2+n3).ne.m) go to 200
          rewind 40
          read(40)rho
          do 201 i=1,81
          rr=r(i)
          do 201 j=1,37
          z=rr*cangl(j)
          xy=rr*sangl(j)
          do 201 k=1,73
          x=xy*cangl(k)
          y=xy*sangl(k)
          if(n1.eq.0)x=1.
          if(n2.eq.0)y=1.
          if(n3.eq.0)z=1.
  201     rho(i,j,k)=rho(i,j,k)*(x**n1)*(y**n2)*(z**n3)
          do 202 k=1,73
          do 202 j=1,37
          ans=0.0
          do 203 i=1,81
  203     ans=ans+rho(i,j,k)*wr(i)
  202     ri1(j,k)=ans
          do 204 k=1,73
          ans=0.0
          do 205 j=1,37
  205     ans=ans+ri1(j,k)*wa1(j)
  204     ri2(k)=ans
          sum=0.0
          do 206 k=1,73
  206     sum=sum+ri2(k)*wa2(k)
          sum=-sum
c
c     *******************************************************************
c
c     add in nuclear part now
c
c     *******************************************************************
c
          do 208 i=1,non
          xnuc=vlist(i,1)**n1
          ynuc=vlist(i,2)**n2
          znuc=vlist(i,3)**n3
          if(n1.eq.0)xnuc=1.
          if(n2.eq.0)ynuc=1.
          if(n3.eq.0)znuc=1.
  208     sum=sum+vlist(4,i)*xnuc*ynuc*znuc
          elmom(m-1,n1+1,n2+1,n3+1)=sum
          write(60,209)n1,n2,n3,sum
  209     format('  nx ',i4,'  ny ',i4,'  nz ',i4,'  monent =  ',f12.6)
  200     continue
  207     continue
      close(unit=60)
c
c     *******************************************************************
c
c     moments up to l=5 have been computed
```

```
c
c     ******************************************************************
c
c
c*********************************************************************
c
c     compute spherical part of the potential
c     get spherically averaged charge density first
c
c*********************************************************************
c
      rewind 40
      read(40) rho
      close(unit=40)
      do 180 i=1,81
      do 180 j=1,37
      ans=0.0
      do 181 k=1,73
 181  ans=ans+rho(i,j,k)*wa2(k)
 180  ri1(i,j)=ans
      do 182 i=1,81
      ans=0.0
      do 183 j=1,37
 183  ans=ans+ri1(i,j)*wa1(j)
 182  ri2(i)=ans
c     spherical rho is in ri2 here
      v1(1)=0.5*r(1)*r(1)*ri2(1)
      v2(1)=0.5*r(1)*ri2(1)
      do 190 i=2,81
      h=(r(i)-r(i-1))*0.5
      a1=(r(i)*r(i)*ri2(i)+r(i-1)*r(i-1)*ri2(i-1))
      a2=(r(i)*ri2(i)+r(i-1)*ri2(i-1))
      v1(i)=v1(i-1)+h*a1
 190  v2(i)=v2(i-1)+h*a2
      do 191 i=1,81
 191  v(i)=v1(i)/r(i)+v2(81)-v2(i)
c     electronic part of the potential determined
c     add in the nuclear part next
      do 192 i=1,non
      do 192 j=1,81
      z1=vlist(i,4)
      r1=vlist(i,1)**2+vlist(i,2)**2+vlist(i,3)**2
      r1=sqrt(r1)
      if(r(j).gt.r1) then
          v(j)=v(j)-z1/r(j)
          else
          v(j)=v(j)-z1/r1
      endif
 192  continue
c     all potential terms are on hand
      open(unit=41,file=mol41(ibb),form='unformatted')
      write(41)xmon,elx,ely,elz,v,elmom
      close(unit=41)
      return
      end
c
c     ******************************************************************
c
c     This subroutine generates the external potential seen by the
c     i th molecular buildingblock. The potential will be generated
c     using a numerical mesh sited about the i th buildingblock.
c     Matrix elements of this potential with the basis vectors
c     on the i th buildingblock will also be evaluated using numerical
c     techniques.
```

```
c
c     written in Fortran 77
c     A B KUNZ at MTU in 1990
c     all rights are reserved by the author
c
c     *****************************************************************
c
      subroutine pot(nbfs,ilps,nenv,xof,yof,zof,a,b,c2,i10,id)
      implicit real*8(a-h,o-z)
      dimension nenv(20),xof(20,200),yof(20,200),zof(20,200),a(20,200),
     1b(20,200),c2(20,200),id(20,200),r(81),v(81,37,73),p1(81,37,73),
     2p2(81,37,73),t1(218781),t2(218781),ri1(37,73),ri2(73),wr(81),wa1(
     337),wa2(73),exv(1024),iii(1024),jjj(1024),u(81),zint(81,37,73),
     4ntype(180),nfirst(180),nlast(180),
     5t3(218781),t4(218781),t5(2701),nr(20,3),eta(
     61024,5),elmom(4,6,6,6),c(1024)
      common/angle/angl(73),cangl(73),sangl(73)
      common/mompot/vlist(1024,4),ntype,nfirst,nlast,eta,c
      integer*2 iii,jjj
      equivalence(p1(1,1,1),t1(1)),(p2(1,1,1),t2(1))
      equivalence(t3(1),v(1,1,1)),(t4(1);zint(1,1,1)),(t5(1),ri1(1,1))
      character*4 zl(20)
      character*11 mol11(20),mol41(20),mol52(20)
      character*15 mol51(20)
      data  nr /  0,1,0,0,2,0,0,1,1,0,3,0,0,2,2,1,0,1,0,1,
     x            0,0,1,0,0,2,0,1,0,1,0,3,0,1,0,2,2,0,1,1,
     x            0,0,0,1,0,0,2,0,1,1,0,0,3,0,1,0,1,2,2,1 /
c
c*********************************************************************
c

      mol51(1)='/work/psi01.dat'
      mol51(2)='/work/psi02.dat'
      mol51(3)='/work/psi03.dat'
      mol51(4)='/work/psi04.dat'
      mol51(5)='/work/psi05.dat'
      mol51(6)='/work/psi06.dat'
      mol51(7)='/work/psi07.dat'
      mol51(8)='/work/psi08.dat'
      mol51(9)='/work/psi09.dat'
      mol51(10)='/work/psi10.dat'
      mol51(11)='/work/psi11.dat'
      mol51(12)='/work/psi12.dat'
      mol51(13)='/work/psi13.dat'
      mol51(14)='/work/psi14.dat'
      mol51(15)='/work/psi15.dat'
      mol51(16)='/work/psi16.dat'
      mol51(17)='/work/psi17.dat'
      mol51(18)='/work/psi18.dat'
      mol51(19)='/work/psi19.dat'
      mol51(20)='/work/psi20.dat'
      mol41(1)='mol4101.dat'
      mol41(2)='mol4102.dat'
      mol41(3)='mol4103.dat'
      mol41(4)='mol4104.dat'
      mol41(5)='mol4105.dat'
      mol41(6)='mol4106.dat'
      mol41(7)='mol4107.dat'
      mol41(8)='mol4108.dat'
      mol41(9)='mol4109.dat'
      mol41(10)='mol4110.dat'
      mol41(11)='mol4111.dat'
      mol41(12)='mol4112.dat'
      mol41(13)='mol4113.dat'
```

```
      mol41(14)='mol4114.dat'
      mol41(15)='mol4115.dat'
      mol41(16)='mol4116.dat'
      mol41(17)='mol4117.dat'
      mol41(18)='mol4118.dat'
      mol41(19)='mol4119.dat'
      mol41(20)='mol4120.dat'
      mol52(1)='mol5201.dat'
      mol52(2)='mol5202.dat'
      mol52(3)='mol5203.dat'
      mol52(4)='mol5204.dat'
      mol52(5)='mol5205.dat'
      mol52(6)='mol5206.dat'
      mol52(7)='mol5207.dat'
      mol52(8)='mol5208.dat'
      mol52(9)='mol5209.dat'
      mol52(10)='mol5210.dat'
      mol52(11)='mol5211.dat'
      mol52(12)='mol5212.dat'
      mol52(13)='mol5213.dat'
      mol52(14)='mol5214.dat'
      mol52(15)='mol5215.dat'
      mol52(16)='mol5216.dat'
      mol52(17)='mol5217.dat'
      mol52(18)='mol5218.dat'
      mol52(19)='mol5219.dat'
      mol52(20)='mol5220.dat'
c
c*********************************************************************
c
c     set up initial integration meshes now
c     linear meshes over angle
c     logarithmic mesh over radius
      open(unit=52,file=mol52(i10),form='unformatted')
      open(unit=51,file=mol51(i10),form='unformatted',
     1access='direct',recl=1750248)
      ilst=1
      ifrst=0
      ibb=i10
      h=-6.9
      del=0.12375
      r(1)=exp(h)
      do 1000 i=3,81,2
      h=h+del+del
      r(i)=exp(h)
 1000 r(i-1)=0.5*(r(i)+r(i-2))
      dang=0.087266463
c     set up integration weight tables
c     angular integrals use weddle's rule on equal integrvals
c     and theta integral is sin(theta) weighted as well
c     radial integrals are using simpson's rule for changing mesh
      wr(1)=(r(2)-r(1))/3.0
      wr(81)=(r(81)-r(80))/3.0
      do 1001 i=2,78,2
      wr(i)=(r(i)-r(i-1))*1.3333333333333
 1001 wr(i+1)=(r(i+2)-r(i))/3.0
      wr(80)=(r(81)-r(80))*1.333333333333
      do 99 i=1,81
   99 wr(i)=wr(i)*r(i)*r(i)
      do 102 i=1,36,6
      wal(i)=2.0
      wal(i+1)=5.0
      wal(i+2)=1.0
      wal(i+3)=6.0
```

```
      wa1(i+4)=1.0
      wa1(i+5)=5.0
 102  continue
      wa1(1)=1.0
      wa1(37)=1.0
      dmul=0.3*dang
      do 103 i=1,37
      th=float(i-1)*dang
 103  wa1(i)=wa1(i)*dmul*sin(th)
      do 104 i=1,72,6
      wa2(i)=2.0
      wa2(i+1)=5.0
      wa2(i+2)=1.0
      wa2(i+3)=6.0
      wa2(i+4)=1.0
      wa2(i+5)=5.0
 104  continue
      wa2(1)=1.0
      wa2(73)=1.0
      do 105 i=1,73
 105  wa2(i)=dmul*wa2(i)
c     integration factors set
c
c***********************************************************************
c
c     all wavefunctions are on file 51 and are direct access
c     form the environmental potential for this buildingblock
c     initially forn v00 part
c     potential is the full potential and includes ionic parts
c
c***********************************************************************
c
      do 120 i=1,81
      do 120 j=1,37
      do 120 k=1,73
 120  v(i,j,k)=0.0
      do 200 ia=1,nenv(i10)
      ib=id(i10,ia)
      x=xof(i10,ia)
      y=yof(i10,ia)
      z=zof(i10,ia)
      open(unit=41,file=mol41(ib),form='unformatted')
      read (41)xion,elx,ely,elz,u,elmom
      close(unit=41)
c
c     ******************************************************************
c
c     do setup for multipole moments here, do rotations of coords
c     go from body coordinates to space coordinates
c     use Euler agles and Cayleigh-Klean parameters
c     actual implementation is only for dipoles
c
c     ******************************************************************
c
      aa=a(i10,ia)
      bb=b(i10,ia)
      ccc=c2(i10,ia)
      sa=sin(aa)
      ca=cos(aa)
      sb=sin(bb)
      cb=cos(bb)
      sc=sin(ccc)
      cc=cos(ccc)
      v11=cc*ca-cb*sa*sc
```

```
      v12=-sc*ca-cb*sa*sc
      v13=sb*sa
      v21=cc*sa+cb*ca*sc
      v22=-sc*sa+cb*ca*cc
      v23=-sb*ca
      v31=sb*sc
      v32=sb*cc
      v33=cb
      px=v11*elx+v12*ely+v13*elz
      py=v21*elx+v22*ely+v23*elz
      pz=v31*elx+v32*ely+v33*elz
c
c     ********************************************************************
c
c     rotated dipoles found
c     can put in rotated higher moments later
c
c     ********************************************************************
c
      zion=u(81)*r(81)
      do 201 i=1,81
 201  u(i)=u(i)-zion/r(i)
c     potential is now missing its ionic tail
      h=-6.9
      del=0.12375
      do 210 i=1,81
      do 210 j=1,37
      do 210 k=1,73
      theta=float(j-1)*dang
      phi=float(k-1)*dang
      xx=r(i)*sin(theta)*cos(phi)
      yy=r(i)*sin(theta)*sin(phi)
      zz=r(i)*cos(theta)
      dx=xx-x
      dy=yy-y
      dz=zz-z
      rr=sqrt(dx*dx+dy*dy+dz*dz)
      v(i,j,k)=v(i,j,k)+xion/rr
      if(i.lt.66.and.rr.lt.r(81))then
          ixa=1+(dlog(rr)-h)/del
          if(ixa.lt.1)ixa=1
          dr=rr-r(ixa)
          da=dr/(r(ixa+1)-r(ixa))
          dv=u(ixa+1)-u(ixa)
          v(i,j,k)=v(i,j,k)+u(ixa)+da*dv
      endif
c
c     *****************************************************************
c
c     add in dipole potential term here
c     later add in higher poles
c
c     *****************************************************************
c
      rsq=rr*rr
      alph=dx/rr
      beta=dy/rr
      gamma=dz/rr
      vd=(px*alph+py*beta+pz*gamma)/rsq
      v(i,j,k)=v(i,j,k)+vd
c
c     *****************************************************************
c
c     dipole potential included
```

```
c     later add higher poles in like way
c
c     *****************************************************************
c
 210  continue
 200  continue
      vmax=0.0
      do 300 i=1,81
      do 300 j=1,37
      do 300 k=1,73
      if (abs(v(i,j,k)).gt.vmax)then
          vmax=abs(v(i,j,k))
          imax=i
          jmax=j
          kmax=k
      endif
 300  continue
      write(61,301)imax,jmax,kmax,r(imax),vmax
 301  format('  POT  VMAX    ',3i5,2f16.4)
      write(61,4589)(v(i,11,3),i=1,81)
 4589 format('  POT  V    ',4f14.4)
c
c*********************************************************************
c
c     potential formed for this case
c     get matrix elements
c
c*********************************************************************
c
      ii=0
      do 220 ia=1,nbfs
      read(51,rec=ia)t1
      do 220 ja=1,ia
      ii=ii+1
      read(51,rec=ja)t2
      do 221 i=1,218781
  221 t4(i)=t1(i)*t2(i)*t3(i)
      do 231 j=1,2701
      ans=0.0
      jof=(j-1)*81
      do 232 i=1,81
  232 ans=ans+t4(jof+i)*wr(i)
  231 t5(j)=ans
      do 233 k=1,73
      ans=0.0
      kof=(k-1)*37
      do 234 j=1,37
  234 ans=ans+t5(kof+j)*wa1(j)
  233 ri2(k)=ans
      sum=0.0
      do 235 k=1,73
 235  sum=sum+ri2(k)*wa2(k)
      exv(ii)=sum
      iii(ii)=ia
      jjj(ii)=ja
      if(ii.eq.1024)then
          write(52)ii,ifrst,iii,jjj,exv
          ii=0
      endif
 220  continue
      write(52)ii,ilst,iii,jjj,exv
      write(61,1289)(exv(i),i=1,300)
 1289 format('  POT  EXV    ',5f12.4)
      close (unit=51)
```

```
      close (unit=52)
      return
      end

c     polyints program -- mqm master file log
c     initial creation -- 10/28/74 -- bdo
c     lppoly local potential integral program - written by cbm 12-1-70
c     added to previous polyatom program
c.....program modified to use ijlk or pair tapes, rvb 09/06/75
c     fortran iv program  pa300 (tape3,tape4,input,output,tape5=input,
c
      subroutine poly(iblk,ilopas,iread,nbfns,non)
      implicit double precision(a-h,o-z)
      real*4tyme(2)
      character*8tlopot
      character*32naml
      character*11 mol11(20),mol5a(20),mol02(20),mol16(20),mol0r(20)
      dimension ncntr(180),ntype(180),nfirst(180),nlast(180),number(180)
     x,mlist(180),vlist(1024,4),icentr(1024),itype(20),nr(20,3),valint(
     y1024)
     x,eta(1024,5),c(1024),nlab(4),kcntr(180),ktype(180)
      dimension mlab(4)
      common/labels/ilbl(18),ilab(18)
      common/ergnuc/energy
      common/inc/x3,x5,x7,x9,x11,x13,x15,x17,x19,x21,x23,x25
      common/ioind/icon(10),ifile2
      common/names/lname(5),iname(5),jname(5)
      common/namtap/nitape,lstnam,notape,intnam,nctape
      common/nmbrs/pi,piterm,pitern,acrcy,scale,icanon
      common/specs/icnt,jcnt,kcnt,lcnt,ityp,jtyp,ktyp,ltyp,is,js,ks,
     1ls,if,jf,kf,lf,m,i,j,k,l
      common/store/str0(280),str1(280),str2(280),str3(280),str4(280),
     1str5(280),str6(280),str7(280),str8(280),str9(280),str10(280),
     2str11(280),str12(280)
      common/lptyp/tlopot(1024)
      common/mompot/vlist,ntype,nfirst,nlast,eta,c
      commonvalint
c     integer rtime
      equivalence(icon1,icon(1)),(icon2,icon(2))
      equivalence(icon10,icon(10))
      data  nr /  0,1,0,0,2,0,0,1,1,0,3,0,0,2,2,1,0,1,0,1,
     x            0,0,1,0,0,2,0,1,0,1,0,3,0,1,0,2,2,0,1,1,
     x            0,0,0,1,0,0,2,0,1,1,0,0,3,0,1,0,1,2,2,1 /
      data  itype/3hs  ,3hx  ,3hy  ,3hz  ,3hxx ,3hyy ,3hzz ,3hxy ,3hxz ,
     x3hyz ,3hxxx,3hyyy,3hzzz,3hxxy,3hxxz,3hxyy,3hyyz,3hxzz,3hyzz,3hxyz/
      data mlab/'slst','tlst','vlst','mlst'/
      data nlab/4hsint,4htint,4hvint,4hmint/,ninmax/1024/,ncmx/1024/,
     xntmx/20/
      mol11(1)='mol1101.dat'
      mol11(2)='mol1102.dat'
      mol11(3)='mol1103.dat'
      mol11(4)='mol1104.dat'
      mol11(5)='mol1105.dat'
      mol11(6)='mol1106.dat'
      mol11(7)='mol1107.dat'
      mol11(8)='mol1108.dat'
      mol11(9)='mol1109.dat'
      mol11(10)='mol1110.dat'
      mol11(11)='mol1111.dat'
      mol11(12)='mol1112.dat'
      mol11(13)='mol1113.dat'
      mol11(14)='mol1114.dat'
      mol11(15)='mol1115.dat'
      mol11(16)='mol1116.dat'
```

```
mol11(17)='mol1117.dat'
mol11(18)='mol1118.dat'
mol11(19)='mol1119.dat'
mol11(20)='mol1120.dat'
mol5a(1)='mol5a01.dat'
mol5a(2)='mol5a02.dat'
mol5a(3)='mol5a03.dat'
mol5a(4)='mol5a04.dat'
mol5a(5)='mol5a05.dat'
mol5a(6)='mol5a06.dat'
mol5a(7)='mol5a07.dat'
mol5a(8)='mol5a08.dat'
mol5a(9)='mol5a09.dat'
mol5a(10)='mol5a10.dat'
mol5a(11)='mol5a11.dat'
mol5a(12)='mol5a12.dat'
mol5a(13)='mol5a13.dat'
mol5a(14)='mol5a14.dat'
mol5a(15)='mol5a15.dat'
mol5a(16)='mol5a16.dat'
mol5a(17)='mol5a17.dat'
mol5a(18)='mol5a18.dat'
mol5a(19)='mol5a19.dat'
mol5a(20)='mol5a20.dat'
mol02(1)='mol0201.dat'
mol02(2)='mol0202.dat'
mol02(3)='mol0203.dat'
mol02(4)='mol0204.dat'
mol02(5)='mol0205.dat'
mol02(6)='mol0206.dat'
mol02(7)='mol0207.dat'
mol02(8)='mol0208.dat'
mol02(9)='mol0209.dat'
mol02(10)='mol0210.dat'
mol02(11)='mol0211.dat'
mol02(12)='mol0212.dat'
mol02(13)='mol0213.dat'
mol02(14)='mol0214.dat'
mol02(15)='mol0215.dat'
mol02(16)='mol0216.dat'
mol02(17)='mol0217.dat'
mol02(18)='mol0218.dat'
mol02(19)='mol0219.dat'
mol02(20)='mol0220.dat'
mol16(1)='mol1601.dat'
mol16(2)='mol1602.dat'
mol16(3)='mol1603.dat'
mol16(4)='mol1604.dat'
mol16(5)='mol1605.dat'
mol16(6)='mol1606.dat'
mol16(7)='mol1607.dat'
mol16(8)='mol1608.dat'
mol16(9)='mol1609.dat'
mol16(10)='mol1610.dat'
mol16(11)='mol1611.dat'
mol16(12)='mol1612.dat'
mol16(13)='mol1613.dat'
mol16(14)='mol1614.dat'
mol16(15)='mol1615.dat'
mol16(16)='mol1616.dat'
mol16(17)='mol1617.dat'
mol16(18)='mol1618.dat'
mol16(19)='mol1619.dat'
mol16(20)='mol1620.dat'
```

```
      mol0r(1)='mol0r01.dat'
      mol0r(2)='mol0r02.dat'
      mol0r(3)='mol0r03.dat'
      mol0r(4)='mol0r04.dat'
      mol0r(5)='mol0r05.dat'
      mol0r(6)='mol0r06.dat'
      mol0r(7)='mol0r07.dat'
      mol0r(8)='mol0r08.dat'
      mol0r(9)='mol0r09.dat'
      mol0r(10)='mol0r10.dat'
      mol0r(11)='mol0r11.dat'
      mol0r(12)='mol0r12.dat'
      mol0r(13)='mol0r13.dat'
      mol0r(14)='mol0r14.dat'
      mol0r(15)='mol0r15.dat'
      mol0r(16)='mol0r16.dat'
      mol0r(17)='mol0r17.dat'
      mol0r(18)='mol0r18.dat'
      mol0r(19)='mol0r19.dat'
      mol0r(20)='mol0r20.dat'
      nfmx=180
      ngmx=1024
      nsavmx=ngmx*(ngmx+1)/2
      nctape=2
      nitape=3
      notape=4
      pi=3.14159265358979d0
      piterm=2.d0/pi**0.5d0
      pitern=pi**1.5d0
      x3 = 1.d0/3.d0
      x5 = 1.d0/5.d0
      x7 = 1.d0/7.d0
      x9 = 1.d0/9.d0
      x11= 1.d0/11.d0
      x13= 1.d0/13.d0
      x15= 1.d0/15.d0
      x17= 1.d0/17.d0
      x19= 1.d0/19.d0
      x21= 1.d0/21.d0
      x23= 1.d0/23.d0
      x25= 1.d0/25.d0
      open(unit=5,file=mol11(iblk),form='formatted')
      open(unit=60,file=mol5a(iblk),form='formatted')
      open(unit=nitape,file=mol02(iblk),
     1form='unformatted')
      open(unit=notape,file=mol16(iblk),
     1form='unformatted')
      open(unit=nctape,file=mol0r(iblk),form='unformatted')
      i=1
      call  rdinpt(nfmx,ngmx,ncmx,ntmx,maxtyp,nbfns,ngaus,noc,
     x                 nlist,c,eta,number,ncntr,ntype,kcntr,ktype,
     x                 nfirst,nlast,mlist,icentr,vlist,itype,nr,non)
      i=10
      if(iread.ne.0)return
      if(icon10.ne.0)nitape=nctape
      read(nitape)ilab
      write(60,7787)ilab
      read(nitape)lbnbf
      if(lbnbf.eq.nbfns)goto7777
      write(60,950)lbnbf,nbfns
      stop
7777  write(notape)ilbl
      if(icon10.eq.0)goto7788
      read(nitape)
```

```
      read(nitape)
      read(nitape)
7788  write(notape)nbfns,(nfirst(i),nlast(i),ncntr(i),ntype(i),
     x kcntr(i),ktype(i),i=1,nbfns)
      write(notape)ngaus,((eta(i,j),j=1,4),c(i),i=1,ngaus)
      write(notape)noc,((vlist(i,j),j=1,4),i=1,noc)
      write(notape)energy,acrcy,scale,icon(9),icanon,maxtyp,nlist,
     x(mlist(i),i=1,nlist)
c.....calculate  the  overlap  integrals  and  check  symmetry
      if(icon2.ge.3)goto540
      i=1
      read(nitape)nnna
      if(nnna.ne.mlab(i).and.nnna.ne.nlab(i))goto959
      write(notape)nlab(i)
540   write (60,912)
      call  gints(ntype,nr,nfirst,nlast,eta,nfmx,ntmx,
     x              ninmax,ngmx,nbfns)
c.....calculate  the  kinetic  energy  integrals
      if(icon2.ge.3)goto550
      i=2
      read(nitape)nnna
      if(nnna.ne.mlab(i).and.nnna.ne.nlab(i))goto959
      write(notape)nlab(i)
550   write (60,913)
      call  tints(ntype,nr,nfirst,nlast,eta,nfmx,ntmx,
     x              ninmax,ngmx)
c.....calculate the potential energy integrals
      if(icon2.ge.3)goto560
      i=3
      read(nitape)nnna
      if(nnna.ne.mlab(i).and.nnna.ne.nlab(i))goto959
      write(notape)nlab(i)
560   write (60,933)
      call  vints(non,vlist,ntype,nr,nfirst,nlast,eta,
     x              nfmx,ncmx,ntmx,ninmax,ngmx)
      if  ( icon(1).eq.1)goto876
c.....calculate the 2-electron integrals
      if(icon2.ge.3)goto570
      i=4
      read(nitape)nnna
      if(nnna.ne.mlab(i).and.nnna.ne.nlab(i))goto959
      write(notape)nlab(i)
  570 if(icon1.lt.2)goto580
      write(60,400)
c.....copy 2-electron integrals
      stop ' no cpycmi '
  580 continue
      write (60,860)
      call  mints(nlist,mlist,ncntr,ntype,eta,nfirst,nlast,
     x              nr,ntmx,ninmax,ngmx,nsavmx,maxtyp,ngaus)
  876 endfilenotape
 772  format(///' time for 1-electron integrals =',f6.2,
     2 ' min.sec, 2-electron integrals =',f6.2,' min.sec')
      go to 901
959   write(60,960)mlab(i),nlab(i),nnna
 901  close(unit=5)
      close(unit=60)
      close(unit=nitape)
      close(unit=notape)
      close(unit=nctape)
 8765 format(1x,f18.4,4x,f18.8)
      return
7787  format('1tape used for labels - ',18a4//)
912   format(' gints - evaluate overlap integrals'//)
```

```
913   format(' tints - evaluate kinetic energy integrals'//)
933   format(' vints - evaluate potential energy integrals'//)
400   format(' cpycmi - copy 2-electron integrals'//)
860   format(' mints - evaluate 2-electron integrals'//)
950   format(//10x,'** labels nbfns =',i4,', does not agree with
     2 integrals nbfns =',i4,' **'/)
960   format(//10x,'** expecting ',a4,' or ',a4,', found ',a4,
     2 ' **'/)
2400  format(1x,'enter the name of the polyin input file (file 5)')
21024  format(a32)
2600  format(1x,'enter the name of the polyin information output
     1 file (file 6)')
2700  format(1x,'enter the name of the labels output file (file 3)')
2800  format(1x,'enter the name of the polyin output file (file 4)')
2900  format(1x,'enter the name of the polyin restart file (file 2)')
      end
      double precision function ovlap (l,m,a,b,t)
      implicit double precision(a-h,o-z)
      ll=l+1
      mm=m+1
      go to (100,101,102,103,104,105),ll
  100 go to (110,111,112,113,200,200),mm
  101 go to (120,121,122,123,200,200),mm
  102 go to (130,131,132,133,200,200),mm
  103 go to (140,141,142,143,200,200),mm
  104 go to (150,151,152,153,200,200),mm
  105 go to (160,161,162,163,200,200),mm
  200 write (60,201)l,m,a,b,t
  201 format (//2i5,3f17.7,15h error in ovlap  )
      stop
c......00
  110 ovlap=1.d0
      go to 300
c......01
  111 ovlap=b
      go to 300
c......02
  112 ovlap=b*b+0.5d0*t
      go to 300
c.....03
  113 ovlap=b*(b*b+1.5d0*t)
      go to 300
c......10
  120 ovlap=a
      go to 300
c......11
  121 ovlap=a*b+0.5d0*t
      go to 300
c......12
  122 ovlap=a*b*b+t*(b+0.5d0*a)
      go to 300
c.....13
  123 ovlap=b*(b*(b*a+1.5d0*t)+1.5d0*a*t)+0.75d0*t*t
      go to 300
c......20
  130 ovlap=a*a+0.5d0*t
      go to 300
c......21
  131 ovlap=a*a*b+t*(a+0.5d0*b)
      go to 300
c......22
  132 ovlap=a*a*b*b+t*(0.5d0*(a*a+b*b)+2.d0*a*b+0.75d0*t)
      go to 300
c.....23
```

```
  133 aa=a*a
      tt=t*t
      ovlap=b*(b*(b*(aa+0.5d0*t)+3.d0*a*t)+1.5d0*aa*t+2.25d0*tt)+1.5d0*a
     x*tt
      go to 300
c......30
  140 ovlap=a*(a*a+1.5d0*t)
      go to 300
c......31
  141 ovlap=a*(a*(a*b+1.5d0*t)+1.5d0*b*t)+0.75d0*t*t
      go to 300
c......32
  142 bb=b*b
      tt=t*t
      ovlap=a*(a*(a*(bb+0.5d0*t)+3.d0*b*t)+1.5d0*bb*t+2.25d0*tt)+1.5d0*b
     x*tt
      go to 300
c.....33
  143 ab=a*b
      abab=a*a+3.d0*ab+b*b
      ovlap=ab*(ab*ab+1.5d0*t*abab)+t*t*(2.25d0*abab+1.875d0*t)
      go to 300
c......40
  150 ovlap=a*a*(a*a+3.d0*t)+0.75d0*t*t
      go to 300
c......41
  151 ovlap=a*(a*(a*(a*b+2.d0*t)+3.d0*b*t)+3.d0*t*t)+0.75d0*b*t*t
      go to 300
c......42
  152 bb=b*b
      tt=t*t
      ovlap=a*(a*(a*(a*(bb+0.5d0*t)+4.d0*b*t)+3.d0*bb*t+4.5d0*tt)+6.d0*b
     x*tt)+     tt*(0.75d0*bb+1.875d0*t)
      go to 300
c.....43
  153 t2=t*t
      b2=b*b
      ovlap=a*(a*(a*(a*(b*(b2+1.5d0*t))+3.d0*t*(2.d0*b2+t))+3.d0*b*t*(b2
     x+4.5d0*t))+t2*(9.d0*b2+7.5d0*t))+b*t2*(0.75d0*b2+5.625d0*t)
      go to 300
c.....50
  160 a2=a*a
      ovlap=a*(5.d0*t*(0.75d0*t+a2)+a2*a2)
      go to 300
c.....51
  161 t2=t*t
      ovlap=a*(a*(a*(a*(a*b+2.5d0*t)+5.d0*b*t)+7.5d0*t2)+3.75d0*b*t2)+1.
     x875d0*t*t2
      go to 300
c.....52
  162 t2=t*t
      b2=b*b
      ovlap=a*(a*(a*(a*(a*(0.5d0*t+b2)+5.d0*b*t)+5.d0*b2*t+7.5d0*t2)+
     x         15.d0*b*t2)+t2*(3.75d0*b2+9.375d0*t))+3.75d0*b*t*t2
      go to 300
c.....53
  163 u=t/2.d0
      u2=u*u
      a2=a*a
      a4=a2*a2
      b2=b*b
      ab=a*b
      ovlap=ab*a4*b2+u*(3.d0*ab*a4+15.d0*a4*b2+10.d0*a2*b2*ab)+3.d0*u2*(
     x         5.d0*a4+30.d0*a2*ab+30.d0*a2*b2+5.d0*ab*b2)+15.d0*u*u2*(10
```

```
     x.d0*a2+15.d0*ab+      3.d0*b2)+105.d0*u2*u2
  300 continue
      return
      end
      double precision function fmch( m,x,y)
      implicit double precision(a-h,o-z)
c        this subroutine evaluates the integral from 0 to 1 of
c                        (u**(2*m)) * expf(-x*(u**2))
c   changes in precision and accuracy made jan.1986 by
c   donald r.beck,mtu. aside from changes in cutoffs
c   to 1.d-15,and accuracy of pi4,and extension of
c   first do loop, argument at which go to asymptotic
c   form was changed from 10 to 20. the asympotic
c   series will not converge to 1.d-15 with arguments
c   between 10 and 20,and so these must be done via
c   the small argument expression.
c   note that the problems of underflow/overflow
c   associated with single precision (hardware)
c   machines like the vax have only been removed
c   for the call from vints. calls from savrge
c   and spdfnt  have still to be explored.
        common/comfmch/pi4,ap0,ap1,ap2,ap3,ap4,ap5,ap6
      if (x-20.d0)10,10,20
   10  a=m
       a=a+0.5d0
       term=1.0d0/a
       ptlsum=term
       do 11 i=2,100
       a=a+1.0d0
       term=term*x/a
       ptlsum=ptlsum+term
       if (term/ptlsum-1.d-15)12,11,11
   11  continue
      write (6,999)m,x
      stop
   12 fmch=0.5d0*ptlsum*y
      go to 150
   20  a=m
      b=a+0.5d0
       a=a-0.5d0
      xd=1.d0/x
      approx=pi4*dsqrt(xd)
       if (m)21,23,21
   21  do 22 i=1,m
       b=b-1.0d0
   22  approx=approx*b
   23 fimult=0.5d0*y*xd
      ptlsum=0.d0
        ap0=approx
        ap1=ap0*xd
        ap2=ap1*xd
        ap3=ap2*xd
        ap4=ap3*xd
        ap5=ap4*xd
        ap6=ap5*xd
        approx=approx*xd**m
c  approx=ap6 only if m=6
c  the call from vints used to multiply approx by x**n
c   (1 .le. n   .le. m)   in the recursion formula.
c by using ap1 through ap6 for large arguments,we have
c actually factored this multiplication into fmch
c  it will only be used by vints if y=0.do (i.e.
c   x .ge.85)
c  the returned value of fmch is the same as it always
```

100

```
c   was
c  note the fmch is used in savrge and spdfnt. the usage
c  in these routines has not been modified as of yet.
      if (fimult)421,25,421
  421 continue
       fiprop=fimult/approx
       term=1.0d0
       ptlsum=term
       notrms=x
       notrms=notrms+m
       do 24 i=2,notrms
       term=term*a*xd
       ptlsum=ptlsum+term
      if (dabs(term*fiprop/ptlsum)-1.d-15)25,25,24
   24  a=a-1.0d0
      write (6,999)m,x
      stop
   25  fmch=approx-fimult*ptlsum
  150 return
  999  format (24h no convergence for fmch, i6, e16.9)
      end
      subroutine   rdinpt (nfmx,ngmx,ncmx,ntmx,maxtyp,nbfns,ngaus,noc,
     $                       nlist,c,eta,number,ncntr,ntype,kcntr,ktype,
     $                       nfirst,nlast,mlist,icntr,vlist,itype,nr,non)
c
c.....icon definitions
c
c     icon(1) - calculate
c         0 = 1e and 2e
c         1 = 1e only
c         2 = 1e and change some 2e (mlist)
c         3 = 1e and copy 2e
c         4 = 1e, copy 2e and add bfns
c         5 = 1e, copy 2e and restart 2e
c
c     icon(2) - tape in
c         0 = none
c         1 = polyatom
c         2 = polyijlk
c         3 = polypair
c         4 = polypair + ijlk
c
c     icon(3) - normalize
c         0 = yes
c         1 = no
c
c     icon(4) - check symmetry
c         0 = yes
c         1 = no
c
      implicit double precision(a-h,o-z)
c
c     valid combinations of icon(1) and icon(2)
c
c                     icon(2)
c             0    1    2    3    4
c     icon(1)
c        0    x    x    x
c        1    x
c        2         x    x
c        3         x    x    x    x
c        4         x    x
c        5         x
c
```

```
      character*8tlopot
      dimension  c(ngmx), eta(ngmx,5), number(nfmx), ncntr(nfmx)
     $         , ntype(nfmx), nfirst(nfmx), nlast(nfmx), mlist(nfmx)
     $         , icntr(ncmx), vlist(ncmx,4), itype(ntmx), nr(ntmx,3)
     $         , kcntr(nfmx),ktype(nfmx)
      common/ergnuc/energy
      common/lptyp/tlopot(1024)
      common/ioind/icon(10),ifile2
      common/labels/ilbl(18),ilab(18)
      common/namtap/nitape,lstnam,notape,intnam
      common/nmbrs/pi,piterm,pitern,acrcy,scale,icanon
      equivalence(icon1,icon(1)),(icon2,icon(2)),(icon3,icon(3)),
     2 (icon4,icon(4)),(icon9,icon(9)),(icon10,icon(10))
      dimension ncon(30)
      dimension wzero(6),zro(6)
      data ncon/1,1,1,0,0,1,0,0,0,0,0,1,1,0,0,1,1,1,1,1,0,1,1,0,0,0,1,0,
     * 0,0/
      data iblnk/4h    /
      data blank/4h    /
      data ijlk,jkpr/'ijlk','jkpr'/
      data zro/' zer','o co','ef s','et t','o on','e   '/
      nlist=0
      icanon=2
      ierr=0
      i=2
c  read and print the problem label.
      read(5,930)(ilbl(ii),ii=1,18)
c  read and print the control options.
      read  (5,900)icon,ifile2
      ifile2=1
      if(icon1.eq.2)icon9=1
      if(icon1.lt.0.or.icon1.gt.5)goto710
      if(icon2.lt.0.or.icon2.gt.4)goto710
      if(ncon(5*icon1+icon2+1).eq.0)goto710
      icon10=icon2
      if(icon1.ge.4)icon10=0
      ilbl(18)=iblnk
      if(icon1.ne.4.and.icon2.eq.2)ilbl(18)=ijlk
      if(icon2.ge.3)ilbl(18)=jkpr
      write(60,931)(ilbl(ii),ii=1,18)
      write (60,901)icon
      go to 670
710   ierr=ierr+1
      write(60,931)(ilbl(ii),ii=1,18)
      write (60,901)icon
      write(60,935)
c  read and print the center coordinates.
670   write(60,671)
      write(60,672)nfmx,ngmx
      write(60,673)ncmx
      write(60,674)
  671 format(//5x,20hprogram limitations      )
  672 format(/5x,25hmax no basis functions=    ,i4
     x       /5x,28hmax no gaussian primitives=     ,i4)
  673 format(/5x,15hmax no centers=  ,i4)
  674 format(/5x,'s,p,d,f type gaussians only')
      read (5,918)non,nac
      write (60,904)non,nac
      noc=non+nac
      if(non.le.ncmx.and.noc.le.ncmx)goto60
        ierr =ierr+1
        write(60,940)
        non=min0(non,ncmx)
        nac=min0(nac,ncmx-non)
```

```
   60 write(60,905)
      do  70i=1,non
        read  (5,906)icntr(i),(vlist(i,j),j=1,4),izz,tlopot(i)
        write (60,907)icntr(i),(vlist(i,j),j=1,4),tlopot(i)
   70 continue
      if  ( nac.le.0)goto90
      k = non+1
      l=non+nac
      write(60,908)
      do  80i=k,l
        read  (5,906)icntr(i),(vlist(i,j),j=1,4)
        write (60,907)icntr(i),(vlist(i,j),j=1,4)
   80   continue
c  read and check the basis functions.
  90  read (5,918)ngaus,nbfns
      write (60,909)ngaus,nbfns
      iq=3
      if  (ngaus.le.ngmx)goto100
        ierr =ierr+1
        write (60,941)
        ngaus =ngmx
 100  if  ( nbfns.le.nfmx)goto110
        ierr =ierr+1
        write (60,942)
        nbfns =nfmx
 110  read  (5,915)(number(i),i=1,nbfns)
      if  ( nbfns.ne.ngaus)goto114
      do  112i=1,nbfns
 112    number(i)=1
 114  nfirst(1)=1
      if(number(1).eq.0)number(1)=1
      nlast(1)=number(1)
      do  120i=2,nbfns
        nfirst(i)=nlast(i-1)+1
      if(number(i).eq.0)number(i)=1
  120   nlast(i)=nlast(i-1)+number(i)
      if  ( nlast(nbfns).eq.ngaus)goto130
        ierr =ierr+1
        write (60,943)
 130 maxtyp=1
      i =  0
      do  300jo=1,nbfns
        isave =0
        isf = number(jo)
       do  290k=1,isf
          i =  i+1
          if  ( isave.ne.0)goto160
         read (5,910) kcnt,ktyp,inc,izz,expnt,c(i)
         kcntr(jo)=kcnt
         ktype(jo)=ktyp
         if  ( inc.eq.0)goto170
         if  ( inc.gt.0.and.inc.lt.jo)goto140
            ierr =ierr+1
            write (60,944) jo,kcnt,ktyp,inc
            go to 300
 140      if  ( number(inc).eq.isf)goto150
            ierr =ierr+1
            write (60,945)jo,inc
            isf = number(inc)
 150      ii =  nfirst(inc)
          isave =1
 160      c(i) =c(ii)
          expnt =eta(ii,4)
          ii =  ii+1
```

```
 170      if  ( expnt.ne.0.d0)goto180
            ierr =ierr+1
            write (60,946)
  180 do 6055mm=1,6
 6055 wzero(mm)=blank
      if(c(i).ne.0.d0)goto190
            c(i) =1.d0
      do 6056mm=1,6
 6056 wzero(mm)=zro(mm)
 190      if  ( k.eq.1.or.ierr.ne.0)goto230
          if(kcnt.eq.icntr(ja)) go to 210
            ierr =ierr+1
            write (60,948)jo,k
210       if(ktyp.eq.itype(jb))goto270
            ierr =ierr+1
            write (60,949)jo,k
          go to 270
 230      do  240jt=1,noc
            ja =  jt
            if(kcnt.eq.icntr(ja)) go to 250
 240        continue
            ierr =ierr+1
            write (60,950)jo
 250      do  260jt=1,ntmx
            jb =  jt
            if(ktyp.eq.itype(jb)) go to 270
 260        continue
            ierr =ierr+1
            write (60,951)jo
 270      ncntr(jo)=ja
          ntype(jo)=jb
      write(60,1911)i,jo,k,kcnt,ktyp,expnt,c(i),(wzero(m),m=1,6)
          do  280m=1,3
 280        eta(i,m)=vlist(ja,m)
          eta(i,4)=expnt
          if  ( jb.gt.maxtyp)maxtyp=jb
 290      continue
 300    continue
      read  (5,912)acrcy,scale
      if  ( acrcy.eq.0.d0)acrcy=1.0d-10
      if  ( scale.eq.0.d0)scale=1.0d0
      scale =scale*acrcy
      write (60,913)acrcy,scale
      if  ( icon(9).ne.1)goto320
      read  (5,918)nlist
      read  (5,918)(mlist(i),i=1,nlist)
      write (60,914)nlist,(mlist(i),i=1,nlist)
      do  310i=1,nlist
c     error in polyatomin next statement-fixed 7/31/69-wyh
c     (nbfn) replaced by (nbfns)
        if  ( mlist(i).gt.0.and.mlist(i).le.nbfns)goto310
        ierr =ierr+1
        write (60,954)
 310    continue
320   if  ( ierr.eq.0)goto330
        write (60,952)ierr
      stop
c  are the basis functions in standard order.
 330  do  340jo=2,nbfns
        if  ( ntype(jo).ge.ntype(jo-1))goto340
        icanon=1
        write (60,922)
        go to 400
 340    continue
```

```
      write (60,923)
  400  continue
c  normalize the primitive functions
      do  420i=1,nbfns
        ityp =ntype(i)
        l =  nr(ityp,1)
        m =  nr(ityp,2)
        n =  nr(ityp,3)
        is =  nfirst(i)
        if =  nlast(i)
        do  410ii=is,if
          t =  0.5d0/eta(ii,4)
          soo = pitern*t**1.5d0
          t1 =  ovlap(l,l,0.0d0,0.0d0,t)
          t2 =  ovlap(m,m,0.0d0,0.0d0,t)
          t3 =  ovlap(n,n,0.0d0,0.0d0,t)
          gii = soo*t1*t2*t3
 410      eta(ii,5)=1.0d0/dsqrt(gii)
 420    continue
      if  ( icon(3).eq.1)goto550
c  renormalize the basis functions.
      write (60,916)
      do  540i=1,nbfns
        ityp =ntype(i)
        l =  nr(ityp,1)
        m =  nr(ityp,2)
        n =  nr(ityp,3)
        is =  nfirst(i)
        if =  nlast(i)
        prtint=0.d0
        do  520ii=is,if
          do  510jj=is,if
            t =  1.0d0/(eta(ii,4)+eta(jj,4))
            soo = pitern*(t**1.5d0)*eta(ii,5)*eta(jj,5)
          t1 =  ovlap(l,l,0.0d0,0.0d0,t)
          t2 =  ovlap(m,m,0.0d0,0.0d0,t)
          t3 =  ovlap(n,n,0.0d0,0.0d0,t)
 510        prtint=prtint+c(ii)*c(jj)*soo*t1*t2*t3
 520      continue
        prtint=1.0d0/dsqrt(prtint)
        do  530k=is,if
          c(k) =c(k)*prtint
          ij =  k-is+1
          write (60,917)k,i,ij,ncntr(i),ntype(i),nr(ityp,1),nr(ityp,2)
     $                  ,nr(ityp,3),eta(k,4),c(k)
 530      continue
 540    continue
 550  do  560k=1,ngaus
 560    eta(k,5)=eta(k,5)*c(k)
c  calculate the nuclear repulsion energy.
      energy=0.d0
      if  ( noc.le.1)goto630
      write (60,919)
      nonml =non-1
      do  620i=1,nonml
        ip1 = i+1
        do  610j=ip1,non
          rij = sqrt((vlist(i,1)-vlist(j,1))**2+(vlist(i,2)
     x              - vlist(j,2))**2 + (vlist(i,3) - vlist(j,3))**2 )
          rija =rij*0.52917d0
          if(non.le.20)write(60,920)icntr(i),icntr(j),rij,rija
          if(rij.lt.1.d-16)go to 899
 610      energy=energy+vlist(i,4)*vlist(j,4)/rij
 620    continue
```

```
 630  write (60,921)energy
      return
899   write (60,955)i,j
      stop
 900  format ( 11i5 )
 901  format ( / 3x, 32hprogram  control  options  ...   ,10i5   )
 904  format ( / 3x, 20hnumber  of  nuclei =, i5, 15x,
     x    33hnumber  of  additional  centers =, i5  )
 905  format ( / 10x, 26h* *  nuclear  centers  * * //3x, 6hcenter, 18x,
     x    11hcoordinates, 21x, 6hcharge,7x,'local potential',/)
  906 format(a4,6x,4f12.8,i2,a8)
 907  format (3x, a4, 6x, 3f12.8, 6x, f12.8 ,5x,a8)
 908  format ( / 10x, 29h* *  additional  centers  * * / 3x, 6hcenter,
     x    18x, 12hcoordinates  / )
 909  format ( // 10x, 44h* *  gaussian  function  specifications  * *
     x    //3x,31hnumber of primitive gaussians =,i5 / 3x,
     x    31hnumber  of  basis  functions  =, i5 //3x,8hgaussian, 3x,
     x    8hfunction, 3x, 9hcomponent, 3x, 6hcenter, 4x, 4htype, 6x,
     x    8hexponent, 6x, 11hcoefficient   )
 910  format ( a4, 6x, a4, i3, i3, 2f12.0 )
 911  format (3(3x,i5,3x), 4x,a4,5x,a4,2f15.7 )
 1911 format( 3(3x,i5,3x), 4x,a4,5x,a4,2f15.7,2x,6a4)
  912 format(2d15.8)
 913  format(//,3x,'dont calculate two-electron integrals',/,
     1' if the prefactor is less than',e15.5,//,3x,
     2'do not write them to disc if they are less than',e15.5)
 914  format ( // 3x,
     x 'integrals which involve the   '  ,i5,
     x10x,24i4)
 915  format ( 36i2 )
 916  format (1h1,10x,  44h* *  renormalize  the  basis  functions  * *
     x    // 3x, 8hgaussian, 3x, 8hfunction, 3x, 9hcomponent, 3x,
     x    6hcenter, 4x, 4htype, 5x, 1hl, 5x, 1hm, 5x, 1hn, 6x,
     x    8hexponent, 6x, 11hcoefficient  )
 917  format ( 3x,i5,6x,i5,6x,i4,5x,i6,4x,i6,1x,3i6,2x,2f15.7 )
 918  format ( 24i3 )
 919  format ( // 10x, 36hinternuclear distances from geometry  //
     x    8x,8h centers,11x, 4ha.u., 10x, 2ha. )
 920  format ( 5x, a4, 3h - , a4, 2f14.6 )
 921  format ( // 3x, 27hnuclear repulsion energy = , f14.8, 6h  a.u.)
 922  format ( /3x,46hthe basis functions are not in standard order )
 923  format ( /3x,49hthe basis functions are listed in standard order )
  930 format(18a4)
  931 format(1h1//5x,18a4 //)
935   format(//10x,'**  incompatible  icon(1)  and  icon(2)  parameters
     2 **'/)
 940  format ( // 10x,26h**  too  many  centers  ** / )
 941  format ( // 10x,29h**  too many  primitives  ** / )
 942  format ( // 10x,28h**  too  many  function  **  / )
 943  format ( // 10x)
 944  format ( // 10x,
     xi4,6x,a4,a4,i4,
     x    4h  ** / )
 945  format ( // 10x,37h**  number of primitives in functions,i5,
     x    4h and,i5,15h  not equal  ** / )
 946  format ( // 10x,22h**  zero  exponent  ** / )
 947  format (    10x,39h**  zero  coefficient  set  to  one  **  )
 948  format ( // 10x,i5,5x,
     xi4/ )
 949  format ( // 10x,35h**  types  not  same  for  function,i5,5x,
     x    10hprimitive ,i4,4h  ** / )
 950  format ( // 10x,36h**  undefined  center  for  function,i5,4h  **)
 951  format ( // 10x,34h**  unallowed  type  for  function,i5,4h  **/)
952   format(//10x,'**',i3,'  error(s).  another  run  for  the  seucr
```

```
     2man  **'/)
 954  format ( // 10x,38h**  undefined  function  in  mlist  ** / )
955   format(1x,'center ',i3,2x,'and center ',i3,2x,'have identical
     1 coordinates')
      end
      subroutine gints (ntype,nr,nfirst,nlast,eta,nfmx,
     $                  ntmx,ninmax,ngmx,nbfns)
      implicit double precision (a-h,o-z)
      integer*2iil(1024),jjl(1024),itgl(1024)
      dimension ntype(nfmx),eta(ngmx,5),nfirst(nfmx),nlast(nfmx),
     1 valint(1024),nr(ntmx,3)
      commonvalint
      dimension s(8256),char(3)
      common/ioind/icon(10)
      common/namtap/nitape,lstnam,notape,intnam
      common/nmbrs/pi,piterm,pitern,acrcy,scale,icanon
      data char/1h ,1h+,1h-/
      ierr =0
      nokk=0
      if(icon(2).ge.3)nokk=1
      if(nokk)3,3,1
    1 kka=0
3     nrcnt=0
      if  ( icon(4).eq.1)goto10
      write (60,992)
      index =nbfns*(nbfns+1)/2
      do  4 i=1,index
   4    s(i) =0.d0
10    if(nokk)11,12,11
11    read(nitape)nints,lstrcd,iil,jjl,itgl
      go to 13
12    read(nitape)nints,lstrcd,iil,jjl,itgl
13    nrcnt=nrcnt+1
      if(nints.le.0)goto1915
      if  ( nints.le.0.or.nints.gt.ninmax)goto800
      do 914m=1,nints
      i=iil(m)
      j=jjl(m)
      itag=itgl(m)
      if  ( icon(4).eq.1)goto550
      index=(i*(i-1))/2+j
      s(index)=1.d0
      go to 403
  550 continue
      if (itag-1)403,402,408
  408 valint(m)=-prvint
      go to 916
  402 valint(m)=prvint
      go to 916
  403 valint(m)=0.d0
      ityp=ntype(i)
      jtyp=ntype(j)
      l1=nr(ityp,1)
      l2=nr(jtyp,1)
      m1=nr(ityp,2)
      m2=nr(jtyp,2)
      n1=nr(ityp,3)
      n2=nr(jtyp,3)
      is=nfirst(i)
      if=nlast(i)
      js=nfirst(j)
      jf=nlast(j)
      do 635ii=is,if
      a=eta(ii,4)
```

```
      do 1635jj=js,jf
      b=eta(jj,4)
      t=1.d0/(a+b)
      p1=(a*eta(ii,1)+b*eta(jj,1))*t
      p2=(a*eta(ii,2)+b*eta(jj,2))*t
      p3=(a*eta(ii,3)+b*eta(jj,3))*t
      ab1=eta(ii,1)-eta(jj,1)
      ab2=eta(ii,2)-eta(jj,2)
      ab3=eta(ii,3)-eta(jj,3)
      distab=ab1*ab1+ab2*ab2+ab3*ab3
      soo=(pitern*t**1.5d0)*exp(-a*b*distab*t)*eta(ii,5)*eta(jj,5)
      pax=p1-eta(ii,1)
      pbx=p1-eta(jj,1)
      pay=p2-eta(ii,2)
      pby=p2-eta(jj,2)
      paz=p3-eta(ii,3)
      pbz=p3-eta(jj,3)
      t1=ovlap(l1,l2,pax,pbx,t)
      t2=ovlap(m1,m2,pay,pby,t)
      t3=ovlap(n1,n2,paz,pbz,t)
 1635 valint(m)=valint(m)+soo*t1*t2*t3
  635 continue
      if  ( icon(4).eq.1)goto510
      if (itag-1)510,511,512
  511 diff=valint(m)-prvint
      go to 513
  512 diff=valint(m)+prvint
  513 if (dabs(diff).lt.1.0d-06)goto916
      write (60,520)ikeep,jkeep,prvint,i,j,char(itag+1),valint(m)
      ierr =ierr+1
      go to 916
  510 prvint=valint(m)
      ikeep=i
      jkeep=j
  916 continue
  914 continue
 1915 if(nokk)1916,1917,1916
 1916 write(notape)nints,lstrcd,iil,jjl,itgl,valint
      go to 1918
 1917 write(notape)nints,lstrcd,iil,jjl,itgl,valint
 1918 if(lstrcd)915,10,915
  915 if  ( icon(4).eq.1)return
      do 581i=1,nbfns
      do 580j=1,i
      index=(i*(i-1))/2+j
      if (s(index).ne.0.d0)goto580
      rawint=0.d0
      ityp=ntype(i)
      jtyp=ntype(j)
      l1=nr(ityp,1)
      l2=nr(jtyp,1)
      m1=nr(ityp,2)
      m2=nr(jtyp,2)
      n1=nr(ityp,3)
      n2=nr(jtyp,3)
      is=nfirst(i)
      if=nlast(i)
      js=nfirst(j)
      jf=nlast(j)
      do 536ii=is,if
      a=eta(ii,4)
      do 535jj=js,jf
      b=eta(jj,4)
      t=1.d0/(a+b)
```

```
      p1=(a*eta(ii,1)+b*eta(jj,1))*t
      p2=(a*eta(ii,2)+b*eta(jj,2))*t
      p3=(a*eta(ii,3)+b*eta(jj,3))*t
      ab1=eta(ii,1)-eta(jj,1)
      ab2=eta(ii,2)-eta(jj,2)
      ab3=eta(ii,3)-eta(jj,3)
      distab=ab1*ab1+ab2*ab2+ab3*ab3
      soo=(pitern*t**1.5d0)*exp(-a*b*distab*t)*eta(ii,5)*eta(jj,5)
      pax=p1-eta(ii,1)
      pbx=p1-eta(jj,1)
      pay=p2-eta(ii,2)
      pby=p2-eta(jj,2)
      paz=p3-eta(ii,3)
      pbz=p3-eta(jj,3)
      t1=ovlap(l1,l2,pax,pbx,t)
      t2=ovlap(m1,m2,pay,pby,t)
      t3=ovlap(n1,n2,paz,pbz,t)
  535 rawint=rawint+soo*t1*t2*t3
  536 continue
      if (dabs(rawint).lt.1.0d-07)goto580
      ierr =ierr+1
      write (60,585)i,j,rawint
  580 continue
  581 continue
      if  ( ierr.eq.0)return
      write (60,993)ierr
      stop
 800  write (60,994)nrcnt,nints
      stop
  520 format (3x, ' symmetry error',5x,'i=',i3,3x,'j=',i3,'prvint=',f14.
     x8,5x,2hi=,i3,3x,2hj=,i3,3x,4htag=,a1,3x,9hintegral=,f14.8)
  585 format (3x,'zero integral',  2i4,3x,'actually is  ' ,f14.8)
992   format(9x,'test symmetry'//)
  993 format(/3x,37h**  gints  cannot  continue ,  ierr =,i5,4h  **)
  994 format(/3x,i5,
     xi10)
      end
      subroutine tints (ntype,nr,nfirst,nlast,eta,nfmx,
     $                    ntmx,ninmax,ngmx)
      implicit double precision (a-h,o-z)
      integer*2iil(1024),jjl(1024),itgl(1024)
      dimension ntype(nfmx),eta(ngmx,5),nfirst(nfmx),nlast(nfmx),
     1 valint(1024),nr(ntmx,3)
      commonvalint
      common/ioind/icon(10)
      common/namtap/nitape,lstnam,notape,intnam
      common/nmbrs/pi,piterm,pitern,acrcy,scale,icanon
      nokk=0
      if(icon(2).ge.3)nokk=1
      if(nokk)3,3,1
 1    do 2 i=1,1024
      iil(i)=0
      jjl(i)=0
 2    itgl(i)=0
3     nrcnt=0
10    if(nokk)11,12,11
11    read(nitape)nints,lstrcd,iil,jjl,itgl
      go to 13
12    read(nitape)nints,lstrcd,iil,jjl,itgl
13    nrcnt=nrcnt+1
      if(nints.le.0)goto1915
      if  ( nints.le.0.or.nints.gt.ninmax)goto800
      do  916m=1,nints
      i=iil(m)
```

```
      j=jj1(m)
      itag=itgl(m)
      if (itag-1)403,402,408
  408 valint(m)=-prvint
      go to 916
  402 valint(m)=prvint
      go to 916
  403 valint(m)=0.d0
      ityp=ntype(i)
      jtyp=ntype(j)
      l1=nr(ityp,1)
      l2=nr(jtyp,1)
      m1=nr(ityp,2)
      m2=nr(jtyp,2)
      n1=nr(ityp,3)
      n2=nr(jtyp,3)
      is=nfirst(i)
      if=nlast(i)
      js=nfirst(j)
      jf=nlast(j)
      do 635ii=is,if
      a=eta(ii,4)
      do 1635jj=js,jf
      b=eta(jj,4)
      t=1.d0/(a+b)
      p1=(a*eta(ii,1)+b*eta(jj,1))*t
      p2=(a*eta(ii,2)+b*eta(jj,2))*t
      p3=(a*eta(ii,3)+b*eta(jj,3))*t
      ab1=eta(ii,1)-eta(jj,1)
      ab2=eta(ii,2)-eta(jj,2)
      ab3=eta(ii,3)-eta(jj,3)
      distab=ab1*ab1+ab2*ab2+ab3*ab3
      soo=(pitern*t**1.5d0)*exp(-a*b*distab*t)*eta(ii,5)*eta(jj,5)
      pax=p1-eta(ii,1)
      pbx=p1-eta(jj,1)
      pay=p2-eta(ii,2)
      pby=p2-eta(jj,2)
      paz=p3-eta(ii,3)
      pbz=p3-eta(jj,3)
      t1=ovlap(l1,l2,pax,pbx,t)
      t2=ovlap(m1,m2,pay,pby,t)
      t3=ovlap(n1,n2,paz,pbz,t)
      s1=ovlap(l2+2,l1,pbx,pax,t)
      s2=ovlap(m2+2,m1,pby,pay,t)
      s3=ovlap(n2+2,n1,pbz,paz,t)
      part=2*(l2+m2+n2)+3
      tke=b*(part*t1*t2*t3-2.d0*b*(s1*t2*t3+t1*s2*t3+t1*t2*s3))
      if (l2-1)190,190,191
  191 part=(l2*(l2-1))/2
      s1=ovlap(l1,l2-2,pax,pbx,t)
      tke=tke-part*s1*t2*t3
  190 if (m2-1)192,192,193
  193 part=(m2*(m2-1))/2
      s2=ovlap(m1,m2-2,pay,pby,t)
      tke=tke-part*t1*s2*t3
  192 if (n2-1)194,194,195
  195 part=(n2*(n2-1))/2
      s3=ovlap(n1,n2-2,paz,pbz,t)
      tke=tke-part*t1*t2*s3
  194 continue
 1635 valint(m)=valint(m)+soo*tke
  635 continue
      prvint=valint(m)
  916 continue
```

```
1915  if(nokk)1916,1917,1916
1916  write(notape)nints,lstrcd,iil,jjl,itgl,valint
      go to 1918
1917  write(notape)nints,lstrcd,iil,jjl,itgl,valint
1918  if(lstrcd)915,10,915
  915 return
  800 write(60,992)nrcnt,nints
      stop
  992 format(/3x,'**  tape read error in tints  nrcnt =',i5,
     x    11h ,  nints =,i10,4h   ** )
      end
      subroutine dawtab
      implicit double precision (a-h,o-z)
c     dawson function generation
      common/dawson/daw(1000)
c     errfun generation
c     y0=err(x), y1=exp(-x**2),y2=-2*x*y1, yn+2=-2*x*yn+1-2*n*yn
      common/errfun/err(550)
      common/dawsf/aoi(50)
      a=0.0d0
      do 5 i=1,50
      a=a+1.0d0
    5 aoi(i)=1.0d0/a
      daw(1)=0.0d0
      h=0.01d0
      nx=999
      h2hm=-h*h*2.0d0
      x2hm=-h2hm
      do 100i=1,nx
      x2hm=x2hm+h2hm
      a0=daw(i)
      a1=h+x2hm*a0
      a2=(h2hm*a0+x2hm*a1)*0.5d0
      a3=(h2hm*a1+x2hm*a2)*0.3333333333333d0
      a4=(h2hm*a2+x2hm*a3)*0.25d0
      a5=(h2hm*a3+x2hm*a4)*0.2d0
  100 daw(i+1)=a0+a1+a2+a3+a4+a5
      h=0.01d0
      x=-0.01d0
      nx=549
      err(1)=0.0d0
      h1o2 =h*0.5d0
      h1o3 =h*0.33333333333333d0
      h1o4 =h*0.25d0
      h1o5 =h*0.20d0
      h1o6 =h/6.0d0
      h1o7 =h/7.0d0
      do 200i=1,nx
      x=x+h
      x2=2.0d0*x
      ex2=exp(-x*x)
      x2m=-x2
      y0=err(i)
      y1=ex2
      y2=x2m*y1
      y3=x2m*y2-2.0d0*y1
      y4=x2m*y3-4.0d0*y2
      y5=x2m*y4-6.0d0*y3
  200 err(i+1)=(y0+h*(y1+h1o2*(y2+h1o3*(y3+h1o4*(y4+
     # h1o5*y5)))))
      return
      end
      subroutine   mints (nlist,mlist,ncntr,ntype,eta,nfirst,nlast,
     $                     nr,ntmx,ninmax,
```

```
     $                    ngmx,nsavmx,maxtyp,ngaus)
      implicit double precision (a-h,o-z)
      integer*2 mu1(1024),mu2(1024),ii1(1024),jj1(1024),kk1(1024),
     1 ll1(1024),
     1 itg1(1024),ii2(1024),jj2(1024),kk2(1024),ll2(1024),itg2(1024)
      doubleprecision s(259560)
      real*4 spval(1024)
      real*8 dpval(1024)
      integer*2 iq1(1024),jq1(1024),kq1(1024)
      dimension ncntr(180),ntype(180),nfirst(180),nlast(180),mlist(180),
     xeta(ngmx,5),nr(ntmx,3),valint(1024),vin(1024)
      common/ioind/icon(10),ifile2
      common/namtap/nitape,lstnam,notape,intnam,nctape
      common/nmbrs/pi,piterm,pitern,acrcy,scale,icanon
      common/specs/icnt,jcnt,kcnt,lcnt,ityp,jtyp,ktyp,ltyp,is,js,ks,
     $               ls,if,jf,kf,lf,nint,i,j,k,l
      commonvalint,vin,ii1,jj1,kk1,ll1,itg1,mu1
      integer*2 iix(1024),jjx(1024),kkx(1024)
      icl=0
c...  generate f integral tables
      call generf(maxtyp,maxrng)
      write(60,1420)maxtyp,maxrng
c...compute and store pre-exponential factor for all i,j index pairs.
      write (60,1410)
      call  savrge(ngaus,eta,s,ngmx,nsavmx)
      icheck=0
      lrecnt=0
      jcan=icanon-1
      nint =0
      lzero =0
      nzrlbl=0
      nzrint=0
      nintot=0
      nlbtot=0
      nogg=0
      ifrst=0
      icntl=0
      ltest1=0
c if ifile2=1, then labels are maintained only in the input file,and
c only the unique integrals are written out onto the old combined file.
c you must save labels output then for the gvb/uhf runs
c this option may be incompatible with the more subtle polyin features,
c like copy,etc.
       if(ifile2.eq.1)write(60,2001)
 2001  format(///5x,'separate files for two electron integrals and
     1  labels--beware')
      if(icon(2).eq.2)nogg=1
      iopt=icon(9)
      if(icon(10).eq.0)goto20
      nitape=3
      nctape=2
      rewindnitape
      read (nitape)
      read (nitape)
      do 201k1=1,3
      read (nitape)
202   read(nitape)ilab,ifml
      if(ifml.eq.0)goto202
201   continue
      read(nitape)
20    if( icon(1).lt.4)goto30
      nint =0
      nlbl=j
      lastrc=k
```

```
      ii=iil(i)
      jj=jjl(i)
      kk=kkl(i)
      ll=lll(i)
      igg=itgl(i)
      muu=mul(i)
      write(60,22)ii,jj,kk,ll
   22 format(1x,'two electron integrals restarted at ',4i5//)
      iml = i-1
      nlbl =nlbl-iml
      do 38 i=1,nlbl
      j = i +iml
      iil(i)=iil(j)
      jjl(i)=jjl(j)
      kkl(i)=kkl(j)
      lll(i)=lll(j)
      itgl(i)=itgl(j)
      mul(i)=mul(j)
   38 continue
      if(ifile2.ne.0)goto236
   13 i = 1
      if( itgl(i)-1)25,26,27
   26 itgl(i)=itgl(i)-1
      go to 25
   27 itgl(i)=itgl(i)-2
   16 i = i +1
      if( i .gt.nlbl)goto200
      if( itgl(i)-1)25,28,29
   28 itgl(i)=itgl(i)+1
      go to 16
   29 itgl(i)=itgl(i)-1
      go to 16
  200 icheck=1
      go to 236
   25 icheck=0
      go to 36
   30 if(iopt)32,341,32
   32 if(ifile2.eq.1)goto341
      read(nctape)nlbl,lastrc,iil,jjl,kkl,lll,itgl,mul,vin
      go to 36
 341  read(nitape)nlbl,lastrc,iil,jjl,kkl,lll,itgl,mul
   36 if( icheck.eq.1)goto13
  236 lrecnt=lrecnt+1
      if(nlbl.le.0)goto700
      if(icon(1).eq.2.and.ifile2.eq.1)goto237
      if(nlbl.gt.ninmax)goto810
237   nlbtot=nlbtot+nlbl
      do  600ij=1,nlbl
      k2=ij
      if(ifile2.eq.0)goto54
      if(iopt.eq.0)goto54
      go to 60
54    if(nogg)55,55,60
55    if(itgl(ij)-1)60,56,58
   56 if(ikp)600,57,600
  57  continue
      go to 600
   58 if(ikp)600,59,600
  59  if(ifile2.eq.1.and.abs(prvint).gt.0.1)then
      icl=icl+1
      dpval(icl)=-prvint
c     iql(icl)=iil(ij)+180*jjl(ij)
c     jql(icl)=kkl(ij)+180*lll(ij)
c     kql(icl)=0+180*mul(ij)
```

```
      endif
      go to 600
   60 i=iil(ij)
      j=jjl(ij)
      k=kkl(ij)
      l=lll(ij)
      itag=itgl(ij)
      nint=nint+1
      ikp=0
      if(iopt)40,80,40
40    if(ifile2.eq.0)goto66
      if(ifrst.eq.0)goto64
      if(icntl.lt.nintl)goto65
64    read(nctape)nintl,ltestl,vin
      ifrst=1
      icntl=0
      ninmax=nintl
65    if(itag.eq.0)goto651
      nint=nint-1
      go to 600
651   icntl=icntl+1
      k2=icntl
      nint=icntl
66    do 50 n=1,nlist
          if  ( mlist(n)-i)44,80,44
 44       if  ( mlist(n)-j)46,80,46
 46       if  ( mlist(n)-k)48,80,48
 48       if  ( mlist(n)-l)50,80,50
 50   continue
      valint(nint)=vin(k2)
      prvint=vin(k2)
      nzrlbl=nzrlbl+1
      nzrint=nzrint+1
       if(ifile2.eq.1)goto2002
      go to 402
   80 nzrlbl=nzrlbl+1
        valint(nint)=0.d0
      if(jcan)120,120,180
 120    if  ( ntype(j)-ntype(i))140,140,130
 130 iii=i
          i  =  j
      j=iii
 140    if  ( ntype(l)-ntype(k))160,160,150
 150 iii=k
          k  =  l
      l=iii
 160  if  ( ntype(k)-ntype(i))180,165,170
 165  if  ( ntype(l)-ntype(j))180,180,170
 170 iii=i
          i  =  k
      k=iii
      iii=j
          j  =  l
      l=iii
 180 icnt=ncntr(i)
      jcnt=ncntr(j)
      kcnt=ncntr(k)
      lcnt=ncntr(l)
      ityp=ntype(i)
      jtyp=ntype(j)
      ktyp=ntype(k)
      ltyp=ntype(l)
      is=nfirst(i)
      js=nfirst(j)
```

```
      ks=nfirst(k)
      ls=nfirst(l)
      if=nlast(i)
      jf=nlast(j)
      kf=nlast(k)
      lf=nlast(l)
      if  ( ityp-4)320,320,321
  320 if (icnt-jcnt)350,351,350
  351 if (kcnt-lcnt)350,352,350
  352 if (icnt-kcnt)350,353,350
 353  call  spones(eta,valint,ngmx,ninmax)
      go to 250
 350  call  spints(eta,valint,s,ngmx,ninmax,nsavmx)
      go to 250
 321  if  ( ityp-10)420,420,421
  421 if (icnt-jcnt)450,451,450
  451 if (kcnt-lcnt)450,452,450
  452 if (icnt-kcnt)450,453,450
 453  call  spdone(eta,valint,s,nr,ntmx,ngmx,ninmax,nsavmx)
      go to 250
 450  call  spdfnt(eta,valint,s,nr,ntmx,ngmx,ninmax,nsavmx)
      go to 250
  420 if (icnt-jcnt)360,361,360
  361 if (kcnt-lcnt)360,362,360
  362 if (icnt-kcnt)360,363,360
 363  call  spdone(eta,valint,s,nr,ntmx,ngmx,ninmax,nsavmx)
      go to 250
 360  call  spdint(eta,valint,s,nr,ntmx,ngmx,ninmax,nsavmx)
  250   prvint=valint(nint)
         nzrint=nzrint+1
 2002      continue
       if(ifile2.ne.1)goto402
       if(dabs(prvint).le.scale)valint(nint)=0.d0
       go to 2009
c  we set all computed integrals below the threshold to zero,to help
c testing in gvb(uhf)
c  all unique integrals must be kept,so that labels tape need not
c  be reset.
  402 if(dabs(prvint)-scale)404,411,411
c   do not write this integral on tape.
 404    nint =nint-1
        nzrint=nzrint-1
      ikp=1
        go to 600
  411 ii2(nint)=ii1(ij)
      jj2(nint)=jj1(ij)
      kk2(nint)=kk1(ij)
      ll2(nint)=ll1(ij)
      itg2(nint)=itg1(ij)
      mu2(nint)=mu1(ij)
 2009    continue
        if  ( nint-ninmax)600,412,412
c   write out an integral record.
 412    irecnt=irecnt+1
        nintot=nintot+nint
      if(ltest1.ne.0)lzero=1
      if(ifile2.ne.1)write(notape)nint,lzero,ii2,jj2,kk2,ll2,itg2,mu2,
     xvalint
      if(ifile2.eq.1)write(notape)nint,lzero,valint
      if(ltest1.ne.0)goto805
        nint =0
 600    continue
 700  if  ( lastrc.eq.0)goto30
        nintot=nintot+nint
```

```
      if(ifile2.ne.1)write(notape)nint,lastrc,ii2,jj2,kk2,ll2,itg2,mu2,
     xvalint
       if(ifile2.eq.1)write(notape)nint,lastrc,valint
      write(60,996)nlbtot,nzrlbl,nintot,nzrint
       if(ifile2.eq.1.and.(.not.(nzrlbl.eq.nintot.and.
     1  nintot.eq.nzrint)))stop'last three quantities not equal-
     2 - as required by ifile2=1'
805   return
  810 write(60,989)lrecnt
      stop
  989 format(/5x,25htoo many labels in record ,i10)
996   format(9x,i7,' labels,',i8,' unique',5x,i7,' integrals written,',
     2 i8,' unique'//)
1420  format(' generf - generate f-integral tables',5x,'maxtyp =',i3,
     2 5x,'maxrng =',i5//)
1410  format(' savrge - compute pre-exponential factors'//)
      end
      subroutine generf (maxtyp,maxrng)
      implicit double precision(a-h,o-z)
      common/inc/x3,x5,x7,x9,x11,x13,x15,x17,x19,x21,x23,x25
      common/store/str0(280),str1(280),str2(280),str3(280),str4(280),
     1str5(280),str6(280),str7(280),str8(280),str9(280),str10(280),
     2str11(280),str12(280)
      maxrng=240
      if (maxtyp-4)762,762,761
  761 maxrng=280
  762 continue
      t=0.d0
      do 779i=1,maxrng
      y=exp(-t)
      u=2.d0*t
      if (maxtyp-4)764,764,765
  765 str12(i)=fmch(12,t,y)
      str11(i)=(u*str12(i)+y)*x23
      str10(i)=(u*str11(i)+y)*x21
      str9(i)=(u*str10(i)+y)*x19
      str8(i)=(u*str9(i)+y)*x17
      go to 766
  764 str8(i)=fmch(8,t,y)
  766 str7(i)=(u*str8(i)+y)*x15
      str6(i)=(u*str7(i)+y)*x13
      str5(i)=(u*str6(i)+y)*x11
      str4(i)=(u*str5(i)+y)*x9
      str3(i)=(u*str4(i)+y)*x7
      str2(i)=(u*str3(i)+y)*x5
      str1(i)=(u*str2(i)+y)*x3
      str0(i)=(u*str1(i)+y)
  779 t=t+0.1d0
      return
      end
      subroutine  savrge (ngaus,eta,s,ngmx,nsavmx)
      implicit double precision(a-h,o-z)
      dimension  eta(ngmx,5), s(nsavmx), ab(3)
      common/nmbrs/pi,piterm,pitern,acrcy,scale
      indx= 0
      do  750ii=1,ngaus
        a= eta(ii,4)
        do  750jj=1,ii
          indx= indx+1
          b= eta(jj,4)
          t1= 1.0d0/(a+b)
          ab(1)=eta(ii,1)-eta(jj,1)
          ab(2)=eta(ii,2)-eta(jj,2)
          ab(3)=eta(ii,3)-eta(jj,3)
```

```
          dab= ab(1)*ab(1)+ab(2)*ab(2)+ab(3)*ab(3)
          s(indx)=pitern*(t1**1.5d0)*exp(-a*b*dab*t1)*eta(ii,5)*eta(jj
     x,5)
 750      continue
      return
      end
      subroutine spones (eta,valint,ngmx,ninmax)
      implicit double precision(a-h,o-z)
      dimension eta(ngmx,5),valint(ninmax)
      common/specs/icnt,jcnt,kcnt,lcnt,ityp,jtyp,ktyp,ltyp,is,js,ks,
     1ls,if,jf,kf,lf,m,i,j,k,l
      common/nmbrs/pi,piterm,pitern,acrcy,scale
      fctrab=1.d0
      fctrcd=1.d0
      do 245ii=is,if
      a=eta(ii,4)
      if (i-j)680,681,680
  681 jf=ii
      fctrab=2.d0
  680 do 244jj=js,jf
      b=eta(jj,4)
      t1=a+b
      if(ii-jj)431,432,431
  432 fctrab=1.d0
  431 do 243kk=ks,kf
      c=eta(kk,4)
      if (k-l)683,682,683
  682 lf=kk
      fctrcd=2.0d0
  683 do 242ll=ls,lf
      d=eta(ll,4)
      if (kk-ll)436,435,436
  435 fctrcd=1.d0
  436 rawint=0.d0
      t2=c+d
      t4=t1+t2
      t1t2=t1*t2
      w=34.9868365d0/dsqrt(t4)
      if (ityp-1)405,406,405
  405 if (ktyp-1)860,831,860
  831 if (ityp-jtyp)242,862,242
  860 if (jtyp-1)870,871,870
  871 if (ltyp-1)870,872,870
  872 if (ityp-ktyp)242,873,242
  870 if (ityp-jtyp)880,881,880
  881 if (ktyp-ltyp)242,883,242
  883 if (ityp-ktyp)884,885,884
  880 if (ityp-ktyp)242,886,242
  886 if (jtyp-ltyp)242,887,242
  406 rawint=w/t1t2
      go to 999
  862 t1t4=t1t2/t4
      t1t4=0.5d0/t1*(1.d0-t1t4/(3.d0*t1))
      rawint=w/t1t2*t1t4
      go to 999
  873 t1t4=0.5d0/(3.d0*t4)
      rawint=w*t1t4/t1t2
      go to 999
  885 t1t4=t1t2/t4
      rawint=(1.d0-(1.d0/3.d0)+3.d0*(t1t4**2)/(5.d0*t1t2))*w/(4.d0*(t1t2
     x)**2)
      go to 999
  884 t1t4=t1t2/t4
      rawint=(1.d0-(1.d0/3.d0)+(t1t4**2)/(5.d0*t1t2))*w/(4.d0*(t1t2)*
```

```
     x*2)
      go to 999
 887 t1t4=t1t2/t4
      rawint=t1t4**2/(5.d0*t1t2)*w/(4.d0*t1t2**2)
 999 rawint=rawint*eta(ii,5)*eta(jj,5)*eta(kk,5)*eta(ll,5)*fctrab*fctrc
     xd
 242 valint(m)=valint(m)+rawint
 243 continue
 244 continue
 245 continue
      return
      end
      subroutine spdone (eta,valint,s,nr,ntmx,ngmx,ninmax,nsavmx)
      implicit double precision(a-h,o-z)
      dimension  f(13),c(13,3),mhi(3),zz(13)
      dimension eta(ngmx,5),s(nsavmx),valint(ninmax),nr(ntmx,3)
      common/specs/icnt,jcnt,kcnt,lcnt,ityp,jtyp,ktyp,ltyp,is,js,ks,
     1ls,if,jf,kf,lf,m,i,j,k,l
      common/nmbrs/pi,piterm,pitern,acrcy,scale
      data f(1)/1.d0/,f(2)/.33333333d0/,f(3)/.2d0/,f(4)/.14285714d0/,f(5
     x)/.11111111d0/,f(6)/.090909091d0/,f(7)/.076923077d0/,f(8)/.0666666
     x67d0/,f(9)/.058823529d0/,f(10)/.052631579d0/,f(11)/.047619047d0/,f
     x(12)/.043478261d0/,f(13)/.04d0/
      fab=1.d0
      fcd=1.d0
      do 353ii=is,if
      a=eta(ii,4)
      if (i-j)680,681,680
 681 jf=ii
      fab=2.d0
 680 do 352jj=js,jf
      b=eta(jj,4)
      t1=a+b
      tab=0.25d0/t1
      index =max0(ii,jj)*(max0(ii,jj)-1)/2+min0(ii,jj)
      saboo=s(index)
      if(ii-jj)431,432,431
 432 fab=1.d0
 431 do 351kk=ks,kf
      cx=eta(kk,4)
      if (k-l)683,682,683
 682 lf=kk
      fcd=2.d0
683  do355ll=ls,lf
      if (kk-ll)436,435,436
 435 fcd=1.d0
 436 continue
      d=eta(ll,4)
      t2=cx+d
      tcd=0.25d0/t2
      prtint=0.d0
      wx=1.d0
      do346n=1,3
      kt=n
      gab=tab
      gcd=tcd
      nij=nr(ityp,kt)+nr(jtyp,kt)+1
      nkl=nr(ktyp,kt)+nr(ltyp,kt)+1
      go to (200,201,202,203,204,205,206),nij
 200 go to (239,499,241,499,242,499,529),nkl
 201 go to (499,243,499,244,499,530,499),nkl
 202 go to (245,499,246,499,247,499,531),nkl
 203 go to (499,249,499,250,499,532,499),nkl
 204 go to (251,499,252,499,254,499,533),nkl
```

```
  205 go to (499,534,499,535,499,536,499),nkl
  206 go to (537,499,538,499,539,499,540),nkl
c......(0)(0)
  239 c(1,kt)=1.d0
      msum=1
      go to 399
c......(0)(2)
  241 c(1,kt)=2.d0*gcd
      c(2,kt)=-2.d0*gcd*gcd
      msum=2
      go to 399
c......(0)(4)
  242 gg=gcd*gcd
      c(1,kt)=12.d0*gg
      c(2,kt)=-24.d0*gcd*gg
      c(3,kt)=12.d0*gg*gg
      msum=3
      go to 399
c......(0)(6)
  529 gcd2=gcd*gcd
      gcd4=gcd2*gcd2
      c(1,kt)=120.d0*gcd*gcd2
      c(2,kt)=-360.d0*gcd4
      c(3,kt)=360.d0*gcd*gcd4
      c(4,kt)=-120.d0*gcd2*gcd4
      msum=4
      go to 399
c......(1)(1)
  243 c(1,kt)=0.d0
      c(2,kt)=2.d0*gab*gcd
      msum=2
      go to 399
c......(1)(3)
  244 gg=gcd*gcd
      c(1,kt)=0.d0
      c(2,kt)=12.d0*gg*gab
      c(3,kt)=-12.d0*gab*gcd*gg
      msum=3
      go to 399
c......(1)(5)
  530 gcd2=gcd*gcd
      gcd4=gcd2*gcd2
      gabcd=gab*gcd
      c(1,kt)=0.d0
      c(2,kt)=120.d0*gabcd*gcd2
      c(3,kt)=-240.d0*gab*gcd4
      c(4,kt)=120.d0*gabcd*gcd4
      msum=4
      go to 399
c......(2)(0)
  245 c(1,kt)=2.d0*gab
      c(2,kt)=-2.d0*gab*gab
      msum=2
      go to 399
c......(2)(2)
  246 gg=gab*gcd
      c(1,kt)=4.d0*gg
      c(2,kt)=-4.d0*gg*(gab+gcd)
      c(3,kt)=12.d0*gg*gg
      msum=3
      go to 399
c......(2)(4)
  247 ggab=gab*gab
      ggcd=gcd*gcd
```

```
      gg=ggab*ggcd
      c(1,kt)=24.d0*ggcd*gab
      c(2,kt)=-24.d0*ggcd*(ggab+2.d0*gab*gcd)
      c(3,kt)=24.d0*(6.d0*gcd*gg+ggcd*ggcd*gab)
      c(4,kt)=-120.d0*gg*ggcd
      msum=4
      go to 399
c......(2)(6)
  531 gcd2=gcd*gcd
      gcd4=gcd2*gcd2
      gabcd=gab*gcd
      gabcd4=gab*gcd4
      c(1,kt)=240.d0*gabcd*gcd2
      c(2,kt)=-240.d0*gab*gcd2*(3.d0*gcd2+gabcd)
      c(3,kt)=720.d0*gabcd4*(3.d0*gab+gcd)
      c(4,kt)=-240.d0*gabcd4*(15.d0*gabcd+gcd2)
      c(5,kt)=1680.d0*gab*gab*gcd2*gcd4
      msum=5
      go to 399
c......(3)(1)
  249 gg=gab*gab
      c(1,kt)=0.d0
      c(2,kt)=12.d0*gg*gcd
      c(3,kt)=-12.d0*gab*gcd*gg
      msum=3
      go to 399
c......(3)(3)
  250 ggab=gab*gab
      ggcd=gcd*gcd
      gg=ggab*ggcd
      c(1,kt)=0.d0
      c(2,kt)=72.d0*gg
      c(3,kt)=-72.d0*gg*(gab+gcd)
      c(4,kt)=120.d0*gg*gab*gcd
      msum=4
      go to 399
c......(3)(5)
  532 gab2=gab*gab
      gcd2=gcd*gcd
      gg=gab2*gcd2*gcd2
      c(1,kt)=0.d0
      c(2,kt)=720.d0*gab2*gcd*gcd2
      c(3,kt)=-720.d0*gab2*gcd2*(2.d0*gcd2+gab*gcd)
      c(4,kt)=240.d0*gg*(10.d0*gab+3.d0*gcd)
      c(5,kt)=-1680.d0*gab*gcd*gg
      msum=5
      go to 399
c......(4)(0)
  251 gg=gab*gab
      c(1,kt)=12.d0*gg
      c(2,kt)=-24.d0*gab*gg
      c(3,kt)=12.d0*gg*gg
      msum=3
      go to 399
c......(4)(2)
  252 ggab=gab*gab
      ggcd=gcd*gcd
      gg=ggab*ggcd
      c(1,kt)=24.d0*ggab*gcd
      c(2,kt)=-24.d0*ggab*(ggcd+2.d0*gab*gcd)
      c(3,kt)=24.d0*(6.d0*gab*gg+ggab*ggab*gcd)
      c(4,kt)=-120.d0*gg*ggab
      msum=4
      go to 399
```

```
c......(4)(4)
  254 ggab=gab*gab
      ggcd=gcd*gcd
      gg=ggab*ggcd
      c(1,kt)=144.d0*gg
      c(2,kt)=-288.d0*gg*(gcd+gab)
      c(3,kt)=144.d0*gg*(ggcd+12.d0*gab*gcd+ggab)
      c(4,kt)=-1440.d0*gg*(gab*ggcd+gcd*ggab)
      c(5,kt)=1680.d0*gg*gg
      msum=5
      go to 399
c......(4)(6)
  533 gab2=gab*gab
      gcd2=gcd*gcd
      gg=gab2*gcd2
      c(1,kt)=1440.d0*gg*gcd
      c(2,kt)=-1440.d0*gg*(3.d0*gcd2+2.d0*gab*gcd)
      c(3,kt)=1440.d0*gg*(3.d0*gcd2*(gcd+6.d0*gab)+gab2*gcd)
      c(4,kt)=-1440.d0*gg*(gcd2*gcd2+30.d0*gab*gcd*gcd2+15.d0*gg)
      c(5,kt)=10080.d0*gg*gcd2*(2.d0*gab*gcd2+5.d0*gcd*gab2)
      c(6,kt)=-30240.d0*gg*gg*gcd2
      msum=6
      go to 399
c......(5)(1)
  534 gab=tcd
      gcd=tab
      go to 530
c......(5)(3)
  535 gab=tcd
      gcd=tab
      go to 532
c......(5)(5)
  536 gab2=gab*gab
      gcd2=gcd*gcd
      gabcd=gab*gcd
      gabcd2=gab2*gcd2
      c(1,kt)=0.d0
      c(2,kt)=7200.d0*gabcd2*gabcd
      c(3,kt)=-14400.d0*gabcd2*(gab*gcd2+gcd*gab2)
      c(4,kt)=2400.d0*gabcd2*(3.d0*gabcd*(gab2+gcd2)+20.d0*gabcd2)
      c(5,kt)=-33600.d0*gabcd2*gabcd2*(gab+gcd)
      c(6,kt)=30240.d0*gabcd2*gabcd2*gabcd
      msum=6
      go to 399
c......(6)(0)
  537 gab=tcd
      gcd=tab
      go to 529
c......(6)(2)
  538 gab=tcd
      gcd=tab
      go to 531
c......(6)(4)
  539 gab=tcd
      gcd=tab
      go to 533
c......(6)(6)
  540 gab2=gab*gab
      gcd2=gcd*gcd
      gabcd=gab*gcd
      gabcd2=gab2*gcd2
      c(1,kt)=14400.d0*gabcd2*gabcd
      c(2,kt)=-43200.d0*gabcd2*(gab*gcd2+gcd*gab2)
      c(3,kt)=43200.d0*gabcd2*(gabcd*(gab2+gcd2)+9.d0*gabcd2)
```

```
      c(4,kt)=-14400.d0*gabcd2*(gab*gcd2*gcd2+45.d0*gabcd2*(gab+gcd)+
     x    gcd*gab2*gab2)
      c(5,kt)=302400.d0*gabcd2*gabcd2*(gab2+5.d0*gabcd+gcd2)
      c(6,kt)=-907200.d0*gabcd2*gabcd2*(gab*gcd2+gcd*gab2)
      c(7,kt)=665280.d0*gabcd2*gabcd2*gabcd2
      msum=7
  399 continue
      mhi(kt)=msum
 346  continue
      mx=mhi(1)
      my=mhi(2)
      mz=mhi(3)
      mxyz=mx+my+mz-2
      z=gab+gcd
      z=1.d0/z
      zz(1)=1.d0
      zz(2)=z
      if (mxyz-2)380,380,382
  382 do 381life=3,mxyz
      nice=life-1
 381  zz(life)=z*zz(nice)
  380 continue
      rawint=0.d0
      do390nx=1,mx
      do389ny=1,my
      do388nz=1,mz
      n=nx+ny+nz-2
 388  rawint=rawint+c(nx,1)*c(ny,2)*c(nz,3)*zz(n)*f(n)
 389  continue
 390  continue
      index =max0(kk,ll)*(max0(kk,ll)-1)/2+min0(kk,ll)
      scdoo=s(index)
      w=0.25d0*z
      wx=piterm*dsqrt(w)*saboo*scdoo*fab*fcd
      prtint=rawint
 355  valint(m)=valint(m)+wx*prtint
 351  continue
 352  continue
 353  continue
  499 continue
      return
       end
      subroutine spints (eta,valint,s,ngmx,ninmax,nsavmx)
      implicit double precision(a-h,o-z)
      dimension eta(ngmx,5),s(nsavmx),valint(ninmax)
      dimension p(3),q(3),r(3),ab(4),cd(3)
      common/store/str0(280),str1(280),str2(280),str3(280),str4(280),
     1str5(280),str6(280),str7(280),str8(280),str9(280),str10(280),
     2str11(280),str12(280)
      common/specs/icnt,jcnt,kcnt,lcnt,ityp,jtyp,ktyp,ltyp,is,js,ks,
     1ls,if,jf,kf,lf,m,i,j,k,l
      common/nmbrs/pi,piterm,pitern,acrcy,scale
      common/inc/x3,x5,x7,x9,x11,x13,x15,x17,x19,x21,x23,x25
      data ab(1)/0.0d00/
      fctrab=1.d0
      fctrcd=1.d0
      addij=0.d0
      addkl=0.d0
      smijoo=0.d0
      smioko=0.d0
      smiool=0.d0
      smojko=0.d0
      smookl=0.d0
      smojol=0.d0
```

```
      smijko=0.d0
      smijol=0.d0
      smiokl=0.d0
      smojkl=0.d0
      msum=ityp+jtyp+ktyp+ltyp-4
      do 350ii=is,if
      a=eta(ii,4)
      if (i-j)680,681,680
  681 jf=ii
      fctrab=2.d0
  680 do 351jj=js,jf
      b=eta(jj,4)
      t1=a+b
      ab(2)=eta(ii,1)-eta(jj,1)
      ab(3)=eta(ii,2)-eta(jj,2)
      ab(4)=eta(ii,3)-eta(jj,3)
      abi=ab(ityp)
      abj=ab(jtyp)
      index =max0(ii,jj)*(max0(ii,jj)-1)/2+min0(ii,jj)
      saboo=s(index)
      if (icnt-jcnt)704,708,704
  704 continue
      p(1)=(a*eta(ii,1)+b*eta(jj,1))/t1
      p(2)=(a*eta(ii,2)+b*eta(jj,2))/t1
      p(3)=(a*eta(ii,3)+b*eta(jj,3))/t1
      go to 709
  708 p(1)=eta(ii,1)
      p(2)=eta(ii,2)
      p(3)=eta(ii,3)
  709 continue
      if (ii-jj)431,432,431
  432 fctrab=1.d0
  431 do 352kk=ks,kf
      c=eta(kk,4)
      if (k-l)683,682,683
  682 lf=kk
      fctrcd=2.d0
  683 do 353ll=ls,lf
      if (kk-ll)436,435,436
  435 fctrcd=1.d0
  436 continue
      d=eta(ll,4)
      t2=c+d
      cd(1)=eta(kk,1)-eta(ll,1)
      cd(2)=eta(kk,2)-eta(ll,2)
      cd(3)=eta(kk,3)-eta(ll,3)
      cdk=cd(ktyp-1)
      cdl=cd(ltyp-1)
      index =max0(kk,ll)*(max0(kk,ll)-1)/2+min0(kk,ll)
      scdoo=s(index)
      t4=t1+t2
      t1t4=t1/t4
      w=t2*t1t4
      wx=piterm*dsqrt(w)*saboo*scdoo*fctrab*fctrcd
      test=2.d0*wx
      if(dabs(test)-acrcy)540,750,750
  750 continue
      t2t4=t2/t4
      if (kcnt-lcnt)705,706,705
  705 continue
      q(1)=(c*eta(kk,1)+d*eta(ll,1))/t2
      q(2)=(c*eta(kk,2)+d*eta(ll,2))/t2
      q(3)=(c*eta(kk,3)+d*eta(ll,3))/t2
      go to 707
```

```
  706 q(1)=eta(kk,1)
      q(2)=eta(kk,2)
      q(3)=eta(kk,3)
  707 continue
      r(1)=p(1)-q(1)
      r(2)=p(2)-q(2)
      r(3)=p(3)-q(3)
      pqsq=r(1)*r(1)+r(2)*r(2)+r(3)*r(3)
      ri=r(ityp-1)
      rj=r(jtyp-1)
      rk=r(ktyp-1)
      rl=r(ltyp-1)
      rawint=0.d0
      if (pqsq)461,461,462
  461 f4=x9
      f3=x7
      f2=x5
      f1=x3
      f0=1.d0
      go to 463
  462 t=w*pqsq
      if (t-23.9d0)4620,4621,4621
 4620 continue
      x=10.d0*(t+0.05d0)
      it= x
      ti=it
      it=it+1
      delt=t-0.1d0*ti
      delt2=0.5d0*delt
      delt3=-delt*x3
c     correction here 8/70-wyh-thd got info from basch
c     delt4=0.25d0*delt
      delt4=-0.25d0*delt
      tf0=str0(it)
      tf1=str1(it)
      tf2=str2(it)
      tf3= str3(it)
      tf4=str4(it)
      if (msum)1462,1462,1463
 1463 continue
      tf5=str5(it)
      tf6=str6(it)
      tf7=str7(it)
      tf8=str8(it)
      f4=tf4+delt*(-tf5+delt2*(tf6+delt3*(tf7+delt4*tf8)))
      f3=tf3+delt*(-tf4+delt2*(tf5+delt3*(tf6+delt4*tf7)))
      f2=tf2+delt*(-tf3+delt2*(tf4+delt3*(tf5+delt4*tf6)))
      f1=tf1+delt*(-tf2+delt2*(tf3+delt3*(tf4+delt4*tf5)))
 1462 continue
      f0=tf0+delt*(-tf1+delt2*(tf2+delt3*(tf3+delt4*tf4)))
      go to 463
 4621 continue
      xd=1.d0/t
      f0=.88622692d0*dsqrt(xd)
      if (msum)463,463,1467
 1467 continue
      f1=0.5d0*xd*f0
      f2=1.5d0*xd*f1
      f3=2.5d0*xd*f2
      f4=3.5d0*xd*f3
  463 continue
      go to (205,202,203,204),ityp
  201 write(60,200)ityp,jtyp,ktyp,ltyp
      stop
```

```
  202 go to (206,207,201,201),jtyp
  203 go to (210,211,212,201),jtyp
  204 go to (214,215,216,217),jtyp
  206 go to (218,422,201,201),ktyp
  207 go to (222,223,201,201),ktyp
  210 go to (218,227,420,201),ktyp
  211 go to (230,231,232,201),ktyp
  212 go to (222,235,236,201),ktyp
  214 go to (218,227,472,419),ktyp
  215 go to (230,231,244,769),ktyp
  216 go to (230,247,248,249),ktyp
  217 go to (222,251,252,253),ktyp
  419 go to (219,470,470,421),ltyp
  420 go to (219,470,421,201),ltyp
  422 go to (219,421,201,201),ltyp
  472 go to (286,387,287,201),ltyp
  223 go to (254,180,201,201),ltyp
  227 go to (286,287,201,201),ltyp
  231 go to (260,261,201,201),ltyp
  232 go to (262,263,987,201),ltyp
  235 go to (264,265,201,201),ltyp
  236 go to (254,283,180,201),ltyp
  244 go to (354,270,271,201),ltyp
  769 go to (262,263,277,987),ltyp
  247 go to (354,271,201,201),ltyp
  248 go to (260,275,261,201),ltyp
  249 go to (262,277,263,987),ltyp
  251 go to (264,265,201,201),ltyp
  252 go to (264,281,265,201),ltyp
  253 go to (254,283,283,180),ltyp
c     ssss
  205 rawint=f0
      go to 999
c     xsss,ysss,zsss
  218 if (abi)710,711,710
  711 rawint=-t2t4*ri*f1
      go to 999
  710 rawint=-b*abi*f0/t1-t2t4*ri*f1
      go to 999
c     xsxs,ysys,zszs
  219 prtint=0.5d0*f1/t4
  319 siooo=-t2t4*ri
      sooko=t1t4*rk
      if (abi)740,741,740
  741 if (cdk)746,742,746
  746 sko=-d*cdk/t2
      rawint=siooo*(sko*f1+sooko*f2)+prtint
      go to 999
  742 rawint=sooko*siooo*f2+prtint
      go to 999
  740 sio=-b*abi/t1
      if (cdk)743,744,743
  744 rawint=sooko*(sio*f1+siooo*f2)+prtint
      go to 999
  743 sko=-d*cdk/t2
      rawint=sko*(sio*f0+siooo*f1)+sooko*(sio*f1+siooo*f2)+prtint
      go to 999
c     xxss,yyss,zzss
  222 siooo=-t2t4*ri
      if (abi)716,717,716
  717 rawint=(f0-t2t4*f1)*0.5d0/t1+f2*siooo**2
      go to 999
  716 continue
      rawint=(f1*(siooo*abi*(a-b)-0.5d0*t2t4)+f0*(0.5d0-a*abi*b*abj/t1))
```

```
     x/tl          + f2*siooo**2
      go to 999
c     ysxs,zsxs,zsys
  286 prtint=0.d0
      go to 319
c     yxss,zxss,zyss
  230 siooo=-t2t4*ri
      sojoo=-t2t4*rj
      if (icnt-jcnt)718,719,718
  719 rawint=f2*siooo*sojoo
      go to 999
  718 continue
      rawint=(f1*(siooo*a*abj-sojoo*b*abi)-a*b*abi*abj*f0/tl)/tl+f2*
     x        siooo*sojoo
      go to 999
c     xsxx,ysyy,zszz
  421 addkl=0.5d0/t2
      smioko=0.5d0*f1/t4
      smiool=smioko
      smookl=-tlt4*f1*addkl
      smiokl=1.5d0*tlt4*ri*f2/t4
  387 sio=-b*abi/tl
      siooo=-t2t4*ri
      sooko=tlt4*rk
      soool=tlt4*rl
      sookl=sooko*soool*f2+smookl
      sioko=siooo*sooko*f2+smioko
      siool=siooo*soool
      siokl=siool*sooko*f3+smiokl
      siool=siool*f2+smiool
      if (kcnt-lcnt)720,721,720
  721 if (addkl)760,761,760
  761 rawint=sio*sookl+siokl
      go to 999
 760  continue
      skl=addkl
      rawint=skl*(sio*f0+siooo*f1)+sio*sookl+siokl
      go to 999
  720 continue
      sko=-d*cdk/t2
      sol=c*cdl/t2
      skl=sko*sol+addkl
      rawint=skl*(sio*f0+siooo*f1)+sko*(sio*soool*f1+siool)+sol*(sio*soo
     x        ko*f1+sioko)+sio*sookl+siokl
      go to 999
c     ysxx,zsxx,zsyy
  287 addkl=0.5d0/t2
      smiokl=0.5d0*tlt4*ri*f2/t4
      smookl=-tlt4*f1*addkl
      go to 387
c     ysyx,zszx,zszy
  470 smioko=0.5d0*f1/t4
      smiokl=0.5d0*tlt4*rl*f2/t4
      go to 387
c     xxxs,yyys,zzzs
  254 addij=0.5d0/tl
      smojko=0.5d0*f1/t4
      smioko=smojko
      smijoo=-t2t4*f1*addij
      smijko=-t2t4*1.5d0*ri*f2/t4
  354 continue
      sko=-d*cdk/t2
      siooo=-t2t4*ri
      sojoo=-t2t4*rj
```

```
      sooko=t1t4*rk
      sijoo=siooo*sojoo*f2+smijoo
      sioko=siooo*sooko*f2+smioko
      sojko=sojoo*sooko
      sijko=sojko*siooo*f3+smijko
      sojko=sojko*f2+smojko
      if (icnt-jcnt)722,723,722
  723 if (addij)765,766,765
  766 rawint=sko*sijoo+sijko
      go to 999
  765 continue
      sij=addij
      rawint=sij*(sko*f0+sooko*f1)+sko*sijoo+sijko
      go to 999
  722 continue
      sio=-b*abi/t1
      soj=a*abj/t1
      sij=sio*soj+addij
      rawint=sij*(sko*f0+sooko*f1)+sio*(sko*sojoo*f1+sojko)+soj*(sko*si
     x       ooo*f1+sioko)+sko*sijoo+sijko
      go to 999
c     yxxs,zxxs,zyys
  260 smojko=0.5d0*f1/t4
      smijko=-t2t4*0.5d0*ri*f2/t4
      go to 354
c     yxys,zxzs,zyzs
  262 smioko=0.5d0*f1/t4
      smijko=-t2t4*0.5d0*rj*f2/t4
      go to 354
c     yyxs,zzxs,zzys
  264 addij=0.5d0/t1
      smijoo=-t2t4*f1*addij
      smijko=-t2t4*0.5d0*rk*f2/t4
      go to 354
c     xxxx,yyyy,zzzz
  180 addij=0.5d0/t1
      addkl=0.5d0/t2
      smojko=0.5d0*f1/t4
      smioko=smojko
      smojol=smojko
      smiool=smojko
      smijoo=-t2t4*f1*addij
      smookl=-t1t4*f1*addkl
      prtint=1.5d0*f2*ri/t4
      smijko=-t2t4*prtint
      smijol=smijko
      smojkl=t1t4*prtint
      smiokl=smojkl
      smijkl=-(3.d0*t1t4*t2t4*ri*rj*f3-0.75d0*f2/t4)/t4
  355 continue
      if (pqsq)777,778,777
  778 sio=-b*abi/t1
      soj= a*abj/t1
      sij= sio*soj+addij
      sko=-d*cdk/t2
      sol= c*cdl/t2
      skl=sko*sol+addkl
      rawint=sij*(skl*f0+smookl)+sio*(sko*smojol+sol*smojko+smojkl)+
     xsoj*(sko*smiool+sol*smioko+smiokl)+skl*smijoo+sko*smijol+
     xsol*smijko+smijkl
      go to 999
  777 continue
      siooo=-t2t4*ri
      sojoo=-t2t4*rj
```

```
      sooko=t1t4*rk
      soool=t1t4*rl
      sijoo=siooo*sojoo
      sookl=soool*sooko
      sijkl=sijoo*sookl*f4+smijkl
      sijko=sijoo*sooko*f3+smijko
      sijol=sijoo*soool*f3+smijol
      sojkl=sojoo*sookl*f3+smojkl
      siokl=siooo*sookl*f3+smiokl
      sijoo=sijoo*f2+smijoo
      sookl=sookl*f2+smookl
      sioko=siooo*sooko*f2+smioko
      siool=siooo*soool*f2+smiool
      sojko=sojoo*sooko*f2+smojko
      sojol=sojoo*soool*f2+smojol
      if (icnt-jcnt)730,731,730
  731 if (kcnt-lcnt)732,733,732
  730 sio=-b*abi/t1
      soj= a*abj/t1
      sij=sio*soj+addij
      if (kcnt-lcnt)734,735,734
  732 continue
      sij=addij
      sko=-d*cdk/t2
      sol= c*cdl/t2
      skl=sko*sol+addkl
      if (sij)770,771,770
  771 rawint=skl*sijoo+sko*sijol+sol*sijko+sijkl
      go to 999
  770 continue
      rawint=sij*(skl*f0+sko*soool*f1+sol*sooko*f1+sookl)+skl*sijoo+
     xsko*sijol+sol*sijko+sijkl
      go to 999
  735 if (addkl)772,773,772
  773 rawint=sij*sookl+sio*sojkl+soj*siokl+sijkl
      go to 999
  772 continue
      skl=addkl
      rawint=sij*(skl*f0+sookl)+sio*(skl*sojoo*f1+sojkl)+soj*(skl*siooo*
     xf1+siokl)+skl*sijoo+sijkl
      go to 999
  733 sij=addij
      skl=addkl
      rawint=sij*(skl*f0+sookl)+skl*sijoo+sijkl
      go to 999
  734 continue
      sko=-d*cdk/t2
      sol= c*cdl/t2
      skl=sko*sol+addkl
      rawint=sij*(skl*f0+sko*soool*f1+sol*sooko*f1+sookl)+sio*(skl*sojoo
     x       *f1+sko*sojol+sol*sojko+sojkl)+soj*(skl*siooo*f1+sko*sioo
     x       l+sol*sioko+siokl)+skl*sijoo+sko*sijol+sol*sijko+sijkl
      go to 999
c     yxxx,zxxx,zyyy
  261 addkl=0.5d0/t2
      smojko=0.5d0*f1/t4
      smookl=-t1t4*f1*addkl
      smojol=smojko
      prtint=0.5d0*f2*ri/t4
      smiokl=t1t4*prtint
      smijko=-t2t4*prtint
      smijol=smijko
      prtint=1.5d0*t1t4*rl/t4
      smojkl=prtint*f2
```

```
      smijkl=-prtint*f3*ri*t2t4
      go to 355
c     yxyy,zxzz,zyzz
  987 addkl=0.5d0/t2
      smioko=0.5d0*f1/t4
      smookl=-t1t4*f1*addkl
      smiool=smioko
      prtint=0.5d0*f2*rj/t4
      smijko=-t2t4*prtint
      smijol=smijko
      smojkl=t1t4*prtint
      prtint=1.5d0*t1t4*ri/t4
      smiokl=prtint*f2
      smijkl=-prtint*f3*rj*t2t4
      go to 355
c     yyyx,zzzx,zzzy
  283 addij=0.5d0/t1
      smojko=0.5d0*f1/t4
      smioko=smojko
      smijoo=-t2t4*f1*addij
      prtint=0.5d0*f2*rl/t4
      smijol=-t2t4*prtint
      smojkl=t1t4*prtint
      smiokl=smojkl
      prtint=1.5d0*t2t4*ri/t4
      smijko=-prtint*f2
      smijkl=-prtint*f3*rl*t1t4
      go to 355
c     yyxx,zzxx,zzyy
  265 addij=0.5d0/t1
      addkl=0.5d0/t2
      smijoo=-t2t4*f1*addij
      smookl=-t1t4*f1*addkl
      prtint=0.5d0*f2/t4
      smijko=-t2t4*prtint*rk
      smijol=smijko
      smojkl=t1t4*prtint*rj
      smiokl=smojkl
      smijkl=-0.5d0*(t1t4*t2t4*f3*(ri*rj+rk*rl)-prtint)/t4
      go to 355
c     yxyx,zxzx,zyzy
  263 smioko=0.5d0*f1/t4
      smojol=smioko
      prtint=0.5d0*f2/t4
      smijko=-t2t4*prtint*rj
      smijol=-t2t4*prtint*ri
      smojkl=t1t4*prtint*rk
      smiokl=t1t4*prtint*rl
      smijkl=-0.5d0*(t1t4*t2t4*f3*(ri*rk+rj*rl)-prtint)/t4
      go to 355
c     zzyx
  281 addij=0.5d0/t1
      smijoo=-t2t4*f1*addij
      prtint=0.5d0*f2/t4*t2t4
      smijko=-prtint*rk
      smijol=-prtint*rl
      smijkl=-0.5d0*t1t4*t2t4*f3*rk*rl/t4
      go to 355
c     zxyx
  270 smojol=0.5d0*f1/t4
      prtint=0.5d0*f2/t4
      smijol=-t2t4*prtint*ri
      smojkl=t1t4*prtint*rk
      smijkl=-0.5d0*t1t4*t2t4*f3*ri*rk/t4
```

```
      go to 355
c     zxyy,zyxx
  271 addkl=0.5d0/t2
      smookl=-tlt4*fl*addkl
      prtint=0.5d0*f2/t4*tlt4
      smojkl=prtint*rj
      smiokl=prtint*ri
      smijkl=-0.5d0*tlt4*t2t4*f3*ri*rj/t4
      go to 355
c     zyyx
  275 smojko=0.5d0*fl/t4
      prtint=0.5d0*f2/t4
      smijko=-t2t4*ri*prtint
      smojkl=tlt4*rl*prtint
      smijkl=-0.5d0*tlt4*t2t4*f3*ri*rl/t4
      go to 355
c     zxzy,  zyzx
  277 smioko=0.5d0*fl/t4
      prtint=0.5d0*f2/t4
      smijko=-t2t4*rj*prtint
      smiokl=tlt4*prtint*rl
      smijkl=-0.5d0*tlt4*t2t4*f3*rj*rl/t4
      go to 355
  999 continue
      rawint=wx*rawint
      valint(m)=valint(m)+rawint
  540 continue
  353 continue
  352 continue
  351 continue
  350 continue
      return
  200 format ( 6h0error,4i4)
      end
      subroutine spdint (eta,valint,s,nr,ntmx,ngmx,ninmax,nsavmx)
      implicit double precision(a-h,c-z)
      dimension eta(ngmx,5),valint(ninmax),s(nsavmx),nr(ntmx,3)
      dimension p(3),q(3),r(3),pa(3),pb(3),qc(3),qd(3),f(13),c(13,3),
     1mhi(3),zz(13),e(7,3),ndex(3)
      common/specs/icnt,jcnt,kcnt,lcnt,ityp,jtyp,ktyp,ltyp,is,js,ks,
     1ls,if,jf,kf,lf,m,i,j,k,l
      common/nmbrs/pi,piterm,pitern,acrcy,scale
      common/gamma/f0,f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,f12
      common/inc/x3,x5,x7,x9,x11,x13,x15,x17,x19,x21,x23,x25
      common/store/str0(280),str1(280),str2(280),str3(280),str4(280),
     $  str5(280),str6(280),str7(280),str8(280),str9(280),str10(280),
     2str11(280),str12(280)
      equivalence(f(1),f0)
      fab=1.d0
      fcd=1.d0
      abxyz=icnt-jcnt
      cdxyz=kcnt-lcnt
      abxyz=dabs(abxyz)
      cdxyz=dabs(cdxyz)
      do 353ii=is,if
      a=eta(ii,4)
      if (i-j)680,681,680
  681 jf=ii
      fab=2.e0
  680 do 352jj=js,jf
      b=eta(jj,4)
      t1=1.d0/(a+b)
      gab=0.25d0*t1
      index =max0(ii,jj)*(max0(ii,jj)-1)/2+min0(ii,jj)
```

```
      saboo=s(index)
      if (abxyz)704,708,704
  704 continue
      p(1)=(a*eta(ii,1)+b*eta(jj,1))*t1
      p(2)=(a*eta(ii,2)+b*eta(jj,2))*t1
      p(3)=(a*eta(ii,3)+b*eta(jj,3))*t1
      go to 709
  708 p(1)=eta(ii,1)
      p(2)=eta(ii,2)
      p(3)=eta(ii,3)
  709 continue
      pa(1)=p(1)-eta(ii,1)
      pa(2)=p(2)-eta(ii,2)
      pa(3)=p(3)-eta(ii,3)
      pb(1)=p(1)-eta(jj,1)
      pb(2)=p(2)-eta(jj,2)
      pb(3)=p(3)-eta(jj,3)
      do 295kt=1,3
      ni=nr(ityp,kt)
      nj=nr(jtyp,kt)
      pakt=pa(kt)
      pbkt=pb(kt)
      if (ni-1)100,101,102
  100 if (nj-1)103,104,105
  101 if (nj-1)106,107,108
  102 if (nj-1)109,110,111
c.....00
  103 index=1
      h10=1.d0
      go to 199
c.....10
  106 h10=pakt
      h11=gab
      index=2
      go to 199
c.....11
 107  index=7
      h12=gab*gab
      if(abxyz)150,151,150
 151  h10=2.d0*gab
      h11=0.d0
      goto199
  150 h10=pakt*pbkt+2.d0*gab
      h11=gab*(pakt+pbkt)
      go to 199
c.....20
 109  index=5
      h12=gab*gab
      if(abxyz)152,153,152
 153  h10=2.d0*gab
      h11=0.d0
      goto199
  152 h10=pakt*pakt+2.d0*gab
      h11=2.d0*gab*pakt
      go to 199
c.....21
  110 gg=gab*gab
      index=14
      h13=gab*gg
      if  (abxyz)154,156,154
  156 h11=6.d0*gg
      h10=0.d0
      h12=0.d0
      goto199
```

```
  154 h10=pakt*pakt*pbkt+2.d0*gab*(2.d0*pakt+pbkt)
      h11=gab*pakt*(2.d0*pbkt+pakt)+6.d0*gg
      h12=gg*(2.d0*pakt+pbkt)
      go to 199
c.....22
  111 gg=gab*gab
      index=22
      h14=gg*gg
      if (abxyz)157,159,157
  159 h12=12.d0*gab*gg
      h10=12.d0*gg
      h11=0.d0
      h13=0.d0
      goto199
 157  ab=pakt*pbkt
      apb=(pakt+pbkt)*(pakt+pbkt)+2.d0*ab
      h10=ab*ab+2.d0*gab*apb+12.d0*gg
      h11=(pakt+pbkt)*(2.d0*gab*ab+12.d0*gg)
      h12=gg*(apb+12.d0*gab)
      h13=2.d0*gg*gab*(pakt+pbkt)
      go to 199
c.....01
  104 h10=pbkt
      h11=gab
      index=2
      go to 199
c.....02
 105  index=5
      h12=gab*gab
      if (abxyz)160,153,160
  160 h10=pbkt*pbkt+2.d0*gab
      h11=pbkt*2.d0*gab
      go to 199
c.....12
  108 gg=gab*gab
      index=14
      h13=gab*gg
      if (abxyz)162,156,162
  162 h10=pbkt*pbkt*pakt+2.d0*gab*(2.d0*pbkt+pakt)
      h11=gab*pbkt*(2.d0*pakt+pbkt)+6.d0*gg
      h12=gg*(2.d0*pbkt+pakt)
  199 continue
      e(1,kt)=h10
      e(2,kt)=h11
      e(3,kt)=h12
      e(4,kt)=h13
      e(5,kt)=h14
      ndex(kt)=index
  295 continue
      if(ii-jj)431,432,431
  432 fab=1.d0
  431 do 351kk=ks,kf
      cx=eta(kk,4)
      if (k-1)683,682,683
  682 lf=kk
      fcd=2.d0
 683  do355ll=ls,lf
      if (kk-ll)436,435,436
  435 fcd=1.d0
  436 continue
      d=eta(ll,4)
      t2=1.e0/(cx+d)
      gcd=0.25e0*t2
      z= gab+gcd
```

```
      z=1.e0/z
      index =max0(kk,ll)*(max0(kk,ll)-1)/2+min0(kk,ll)
      scdoo=s(index)
      w=0.25e0*z
      wx=piterm*dsqrt(w)*saboo*scdoo*fab*fcd
      test=2.e0*wx
      prtint=0.e0
      if (dabs(test)-acrcy)540,750,750
  750 if (cdxyz)705,706,705
  705 continue
      q(1)=(cx*eta(kk,1)+d*eta(ll,1))*t2
      q(2)=(cx*eta(kk,2)+d*eta(ll,2))*t2
      q(3)=(cx*eta(kk,3)+d*eta(ll,3))*t2
      go to 707
  706 q(1)=eta(kk,1)
      q(2)=eta(kk,2)
      q(3)=eta(kk,3)
  707 r(1)=p(1)-q(1)
      r(2)=p(2)-q(2)
      r(3)=p(3)-q(3)
      pqsq=r(1)*r(1)+r(2)*r(2)+r(3)*r(3)
      qc(1)=q(1)-eta(kk,1)
      qc(2)=q(2)-eta(kk,2)
      qc(3)=q(3)-eta(kk,3)
      qd(1)=q(1)-eta(ll,1)
      qd(2)=q(2)-eta(ll,2)
      qd(3)=q(3)-eta(ll,3)
      rawint=0.d0
      do 346kt=1,3
      ni=nr(ityp,kt)
      nj=nr(jtyp,kt)
      nk=nr(ktyp,kt)
      nl=nr(ltyp,kt)
      msum=ni+nj+nk+nl+1
      if (msum-1)2,2,610
  610 continue
      qckt=qc(kt)
      qdkt=qd(kt)
      rkt=-r(kt)
      h10=e(1,kt)
      h11=e(2,kt)
      h12=e(3,kt)
      h13=e(4,kt)
      h14=e(5,kt)
      index=ndex(kt)
      if(nk-1)200,201,202
  200 if(nl-1)203,204,205
  201 if (nl-1)206,207,208
  202 if(nl-1)209,210,211
c.....00
  203 index=index+1
      hm0=1.d0
      goto299
c.....10
  206 hm0=qckt
      hm1=-gcd
      index=index+2
      goto299
c.....11
 207  index=index+7
      hm2=gcd*gcd
      if(cdxyz)250,251,250
 251  hm0=2.d0*gcd
      hm1=0.d0
```

```
      goto299
  250 hm0=qckt*qdkt+2.d0*gcd
      hm1=-gcd*(qckt+qdkt)
      goto299
c.....20
 209  index=index+5
      hm2=gcd*gcd
      if(cdxyz)252,253,252
 253  hm0=2.d0*gcd
      hm1=0.d0
      goto299
  252 hm0=qckt*qckt+2.d0*gcd
      hm1=-2.d0*gcd*qckt
      goto299
c.....21
  210 gg=gcd*gcd
      index=index+14
      hm3=-gcd*gg
      if(cdxyz)254,256,254
  256 hm1=-6.d0*gg
      hm0=0.d0
      hm2=0.d0
      goto299
  254 hm0=qckt*qckt*qdkt+2.d0*gcd*(2.d0*qckt+qdkt)
      hm1=-(gcd*qckt*(2.d0*qdkt+qckt)+6.d0*gg)
      hm2=gg*(2.d0*qckt+qdkt)
      goto299
c.....22
  211 gg=gcd*gcd
      index=index+22
      hm4=gg*gg
      if(cdxyz)257,259,257
  259 hm2=12.d0*gcd*gg
      hm0=12.d0*gg
      hm1=0.d0
      hm3=0.d0
      goto299
  257 cd=qckt*qdkt
      cpd=(qckt+qdkt)*(qckt+qdkt)+2.d0*cd
      hm0=cd*cd+2.d0*gcd*cpd+12.d0*gg
      hm1=-(qckt+qdkt)*(2.d0*gcd*cd+12.d0*gg)
      hm2=gg*(cpd+12.d0*gcd)
      hm3=-2.d0*gg*gcd*(qckt+qdkt)
      goto299
c.....01
  204 hm0=qdkt
      hm1=-gcd
      index=index+2
      goto299
c.....02
 205  index=index+5
      hm2=gcd*gcd
      if (cdxyz)260,253,260
  260 hm0=qdkt*qdkt+2.d0*gcd
      hm1=-2.d0*gcd*qdkt
      goto299
c.....12
  208 gg=gcd*gcd
      index=index+14
      hm3=-gcd*gg
      if (cdxyz)262,256,262
  262 hm0=qdkt*qdkt*qckt+2.d0*gcd*(2.d0*qdkt+qckt)
      hm1=-(gcd*qdkt*(2.d0*qckt+qdkt)+6.d0*gg)
      hm2=gg*(2.d0*qdkt+qckt)
```

```
  299 continue
      goto(1,2,3,4,1,6,7,6,7,10,1,10,1,10,15,16,1,1,19,1,19,1,23,24,1,1,
     127,28,27,1,1,1,1,1,1,36,1,1,1,1,1,1,1,44),index
 1    write(60,298)index,i,j,k,l,ii,jj,kk,ll,ni,nj,nk,nl,kt
 298  format(i5,13i5)
      stop
c.....0000
 2    c(1,kt)=1.d0
      go to 399
c.....1000 0100 0010 0001
 3    if(ni+nj)1,310,311
 311  c(1,kt)=hl0
      c(2,kt)=rkt*hl1
      go to 399
 310  c(1,kt)=hm0
      c(2,kt)=rkt*hm1
      go to 399
cc....1010 1001 0110 0101
 4    hlm1=hl1*hm0+hl0*hm1
      hlm2=hl1*hm1
 312  c(1,kt)=hl0*hm0
      c(2,kt)=rkt*hlm1-2.d0*hlm2
      c(3,kt)=rkt*rkt*hlm2
      go to 399
c.....2000 0200 0020 0002 1100 0011
 6    if(ni+nj)1,313,309
 313  hlm1=hm1
      hlm2=hm2
      goto312
 309  hlm1=hl1
      hlm2=hl2
      goto312
c.....2010 2001 0210 0201 1020 1002 0120 0102 1110 1101 1011 0111
 7    if(ni+nj-nk-nl)314,1,315
 314  hlm2=hl1*hm1+hl0*hm2
      hlm3=hl1*hm2
      goto3080
 315  hlm2=hl1*hm1+hl2*hm0
      hlm3=hl2*hm1
 3080 hlm1=hl1*hm0+hl0*hm1
 308  rkt2=rkt*rkt
      rkt3=rkt*rkt2
      c(1,kt)=hl0*hm0
      c(2,kt)=rkt*hlm1-2.d0*hlm2
      c(3,kt)=rkt2*hlm2-6.d0*hlm3*rkt
      c(4,kt)=rkt3*hlm3
      go to 399
c.....2020 2002 0220 0202 2011 0211 1120 1102 1111
 10   hlm1=hl0*hm1+hl1*hm0
      hlm2=hl0*hm2+hl1*hm1+hl2*hm0
      hlm3=hl2*hm1+hl1*hm2
      hlm4=hl2*hm2
  307 if (rkt)372,371,372
  372 if (abxyz+cdxyz)376,376,377
 376  rkt2=rkt*rkt
      rkt4=rkt2*rkt2
      c(1,kt)=hl0*hm0
      c(2,kt)=-2.d0*hlm2
      c(3,kt)=rkt2*hlm2+12.d0*hlm4
      c(4,kt)=-12.d0*rkt2*hlm4
      c(5,kt)=rkt4*hlm4
      goto399
 377  continue
      rkt2=rkt*rkt
```

```
      rkt3=rkt*rkt2
      rkt4=rkt*rkt3
      c(1,kt)=hl0*hm0
      c(2,kt)=rkt*hlm1-2.d0*hlm2
      c(3,kt)=rkt2*hlm2-6.d0*rkt*hlm3+12.d0*hlm4
      c(4,kt)=rkt3*hlm3-12.d0*rkt2*hlm4
      c(5,kt)=rkt4*hlm4
      go to 399
c.....2100 1200 0021 0012
 15   if(ni)1,316,317
 316  hlm1=hm1
      hlm2=hm2
      hlm3=hm3
      goto308
 317  hlm1=hl1
      hlm2=hl2
      hlm3=hl3
      goto308
c.....2110 2101 1210 1201 1021 1012 0121 0112
 16   if(ni+nj-nk-nl)318,1,319
 318  hlm1=hl0*hm1+hl1*hm0
      hlm2=hl0*hm2+hl1*hm1
      hlm3=hl1*hm2+hl0*hm3
      hlm4=hl1*hm3
      goto307
 319  hlm1=hl0*hm1+hl1*hm0
      hlm2=hl2*hm0+hl1*hm1
      hlm3=hl2*hm1+hl3*hm0
      hlm4=hl3*hm1
      go to 307
c.....2120 2102 1220 1202 2021 2012 0221 0212 2111 1211 1121 1112
 19   if(ni+nj-nk-nl)320,1,321
 320  hlm1=hl0*hm1+hl1*hm0
      hlm2=hl0*hm2+hl1*hm1+hl2*hm0
      hlm3=hl1*hm2+hl0*hm3+hl2*hm1
      hlm4=hl2*hm2+hl1*hm3
      hlm5=hl2*hm3
      goto322
 321  hlm1=hl0*hm1+hl1*hm0
      hlm2=hl0*hm2+hl1*hm1+hl2*hm0
      hlm3=hl2*hm1+hl3*hm0+hl1*hm2
      hlm4=hl2*hm2+hl3*hm1
      hlm5=hl3*hm2
  322 if (rkt)370,371,370
 371  c(1,kt)=hl0*hm0
      c(2,kt)=-2.d0*hlm2
      c(3,kt)=12.d0*hlm4
      msum=3
      goto399
  370 if (abxyz+cdxyz)1372,1372,373
 1372 rkt2=rkt*rkt
      rkt3=rkt*rkt2
      rkt5=rkt2*rkt3
      c(1,kt)=0.d0
      c(2,kt)=rkt*hlm1
      c(3,kt)=-6.d0*rkt*hlm3
      c(4,kt)=rkt3*hlm3+60.d0*rkt*hlm5
      c(5,kt)=-20.d0*rkt3*hlm5
      c(6,kt)=rkt5*hlm5
      goto399
 373  continue
      rkt2=rkt*rkt
      rkt3=rkt*rkt2
      rkt4=rkt*rkt3
```

```
      rkt5=rkt*rkt4
      c(1,kt)=h10*hm0
      c(2,kt)=rkt*hlm1-2.d0*hlm2
      c(3,kt)=rkt2*hlm2-6.d0*rkt*hlm3+12.d0*hlm4
      c(4,kt)=rkt3*hlm3-12.d0*rkt2*hlm4+60.d0*rkt*hlm5
      c(5,kt)=rkt4*hlm4-20.d0*rkt3*hlm5
      c(6,kt)=rkt5*hlm5
      go to 399
c.....2200 0022
 23   if(ni)1,323,324
 323  hlm1=hm1
      hlm2=hm2
      hlm3=hm3
      hlm4=hm4
      goto307
 324  hlm1=hl1
      hlm2=hl2
      hlm3=hl3
      hlm4=hl4
      goto307
c.....2210 2201 1022 0122
 24   if(ni-nj)325,326,325
 325  hlm1=h10*hm1+hl1*hm0
      hlm2=hl1*hm1+h10*hm2
      hlm3=hl1*hm2+h10*hm3
      hlm4=hl1*hm3+h10*hm4
      hlm5=hl1*hm4
      goto322
 326  hlm1=h10*hm1+hl1*hm0
      hlm2=hl1*hm1+hl2*hm0
      hlm3=hl2*hm1+hl3*hm0
      hlm4=hl3*hm1+hl4*hm0
      hlm5=hl4*hm1
      goto322
c.....2220 2202 2022 0222 2211 1122
 27   hlm1=h10*hm1+hl1*hm0
      hlm2=h10*hm2+hl1*hm1+hl2*hm0
      if (ni+nj-nk-nl)328,1,329
 328  hlm3=hm1*hl2+hm2*hl1+hm3*h10
      hlm4= hm2*hl2+hm3*hl1+hm4*h10
      hlm5=hm3*hl2+hm4*hl1
      hlm6=hm4*hl2
      goto330
 329  hlm3=hl1*hm2+hl2*hm1+hl3*hm0
      hlm4=hl2*hm2+hl3*hm1+hl4*hm0
      hlm5=hl3*hm2+hl4*hm1
      hlm6=hl4*hm2
  330 if (rkt)360,1350,360
  360 if (abxyz+cdxyz)1352,1352,1353
 1352 rkt2=rkt*rkt
      rkt4=rkt2*rkt2
      rkt6=rkt2*rkt4
      c(1,kt)=h10*hm0
      c(2,kt)=-2.d0*hlm2
      c(3,kt)=rkt2*hlm2+12.d0*hlm4
      c(4,kt)=-12.d0*rkt2*hlm4-120.d0*hlm6
      c(5,kt)=rkt4*hlm4+180.d0*rkt2*hlm6
      c(6,kt)=-30.d0*rkt4*hlm6
      c(7,kt)=rkt6*hlm6
      goto399
 1353 continue
      rkt2=rkt*rkt
      rkt3=rkt*rkt2
      rkt4=rkt*rkt3
```

```
      rkt5=rkt*rkt4
      rkt6=rkt*rkt5
      c(1,kt)=h10*hm0
      c(2,kt)=rkt*hlm1-2.d0*hlm2
      c(3,kt)=rkt2*hlm2-6.d0*rkt*hlm3+12.d0*hlm4
      c(4,kt)=rkt3*hlm3-12.d0*rkt2*hlm4+60.d0*rkt*hlm5-120.d0*hlm6
      c(5,kt)=rkt4*hlm4-20.d0*rkt3*hlm5+180.d0*rkt2*hlm6
      c(6,kt)=rkt5*hlm5-30.d0*rkt4*hlm6
      c(7,kt)=rkt6*hlm6
      go to 399
c.....2121 2112 1221 1212
 28   hlm1=h10*hm1+h11*hm0
      hlm2=h10*hm2+h11*hm1+h12*hm0
      hlm3=h10*hm3+h11*hm2+h12*hm1+h13*hm0
      hlm4=h11*hm3+h12*hm2+h13*hm1
      hlm5=h12*hm3+h13*hm2
      hlm6=h13*hm3
      goto330
c.....2221 2212 2122 1222
 36   hlm1=h11*hm0+h10*hm1
      hlm2=h10*hm2+h11*hm1+h12*hm0
      hlm3=h10*hm3+h11*hm2+h12*hm1+h13*hm0
      if(ni-nj)331,332,331
 331  hlm4=hm1*h13+hm2*h12+hm3*h11+hm4*h10
      hlm5=hm2*h13+hm3*h12+hm4*h11
      hlm6=hm3*h13+hm4*h12
      hlm7=hm4*h13
      goto333
 332  hlm4=h11*hm3+h12*hm2+h13*hm1+h14*hm0
      hlm5=h12*hm3+h13*hm2+h14*hm1
      hlm6=h13*hm3+h14*hm2
      hlm7=h14*hm3
  333 if (rkt)1351,1350,1351
 1350 c(1,kt)=h10*hm0
      c(2,kt)=-2.d0*hlm2
      c(3,kt)=12.d0*hlm4
      c(4,kt)=-120.d0*hlm6
      msum=4
      go to 399
 1351 if (abxyz+cdxyz)366,366,367
 366  rkt2=rkt*rkt
      rkt3=rkt*rkt2
      rkt5=rkt2*rkt3
      rkt7=rkt2*rkt5
      c(1,kt)=0.d0
      c(2,kt)=rkt*hlm1
      c(3,kt)=-6.d0*rkt*hlm3
      c(4,kt)=rkt3*hlm3+60.d0*rkt*hlm5
      c(5,kt)=-20.d0*rkt3*hlm5-840.d0*rkt*hlm7
      c(6,kt)=rkt5*hlm5+420.d0*rkt3*hlm7
      c(7,kt)=-42.d0*rkt5*hlm7
      c(8,kt)=rkt7*hlm7
      goto399
 367  continue
      rkt2=rkt*rkt
      rkt3=rkt*rkt2
      rkt4=rkt*rkt3
      rkt5=rkt*rkt4
      rkt6=rkt*rkt5
      rkt7=rkt*rkt6
      c(1,kt)=h10*hm0
      c(2,kt)=rkt*hlm1-2.d0*hlm2
      c(3,kt)=rkt2*hlm2-6.d0*rkt*hlm3+12.d0*hlm4
      c(4,kt)=rkt3*hlm3-12.d0*rkt2*hlm4+60.d0*rkt*hlm5-120.d0*hlm6
```

```
      c(5,kt)=rkt4*hlm4-20.d0*rkt3*hlm5+180.d0*rkt2*hlm6-840.d0*rkt*hlm7
      c(6,kt)=rkt5*hlm5-30.d0*rkt4*hlm6+420.d0*rkt3*hlm7
      c(7,kt)=rkt6*hlm6-42.d0*rkt5*hlm7
      c(8,kt)=rkt7*hlm7
      goto399
c.....2222
 44   hlm1=hl0*hm1+hl1*hm0
      hlm2=hl0*hm2+hl1*hm1+hl2*hm0
      hlm3=hl0*hm3+hl1*hm2+hl2*hm1+hl3*hm0
      hlm4=hl0*hm4+hl1*hm3+hl2*hm2+hl3*hm1+hl4*hm0
      hlm5=hl1*hm4+hl2*hm3+hl3*hm2+hl4*hm1
      hlm6=hl2*hm4+hl3*hm3+hl4*hm2
      hlm7=hl3*hm4+hl4*hm3
      hlm8=hl4*hm4
      if (rkt)363,362,363
 362  c(1,kt)=hl0*hm0
      c(2,kt)=-2.d0*hlm2
      c(3,kt)=12.d0*hlm4
      c(5,kt)=1680.d0*hlm8
      msum=5
      c(4,kt)=-120.d0*hlm6
      goto399
 363  if(abxyz+cdxyz)364,364,365
 364  rkt2=rkt*rkt
      rkt4=rkt2*rkt2
      rkt6=rkt2*rkt4
      rkt8=rkt4*rkt4
      c(1,kt)=hl0*hm0
      c(2,kt)=-2.d0*hlm2
      c(3,kt)=rkt2*hlm2+12.d0*hlm4
      c(4,kt)=-12.d0*rkt2*hlm4-120.d0*hlm6
      c(5,kt)=rkt4*hlm4+180.d0*rkt2*hlm6+1680.d0*hlm8
      c(6,kt)=-30.d0*rkt4*hlm6-3360.d0*rkt2*hlm8
      c(7,kt)=rkt6*hlm6+840.d0*rkt4*hlm8
      c(8,kt)=-56.d0*rkt6*hlm8
      c(9,kt)=rkt8*hlm8
      goto399
 365  rkt2= rkt*rkt
      rkt3=rkt*rkt2
      rkt4=rkt*rkt3
      rkt5=rkt*rkt4
      rkt6=rkt*rkt5
      rkt7=rkt*rkt6
      rkt8=rkt*rkt7
      c(1,kt)=hl0*hm0
      c(2,kt)=rkt*hlm1-2.d0*hlm2
      c(3,kt)=rkt2*hlm2-6.d0*rkt*hlm3+12.d0*hlm4
      c(4,kt)=rkt3*hlm3-12.d0*rkt2*hlm4+60.d0*rkt*hlm5-120.d0*hlm6
      c(5,kt)=rkt4*hlm4-20.d0*rkt3*hlm5+180.d0*rkt2*hlm6-840.d0*rkt*hlm7
     x+1680.d0*hlm8
      c(6,kt)=rkt5*hlm5-30.d0*rkt4*hlm6+420.d0*rkt3*hlm7-3360.d0*rkt2*hl
     xm8
      c(7,kt)=rkt6*hlm6-42.d0*rkt5*hlm7+840.d0*rkt4*hlm8
      c(8,kt)=rkt7*hlm7-56.d0*rkt6*hlm8
      c(9,kt)=rkt8*hlm8
 399  continue
      mhi(kt)=msum
 346  continue
      mx=mhi(1)
      my=mhi(2)
      mz=mhi(3)
      mxyz=mx+my+mz-2
      if (pqsq)461,461,462
 461  f8=x17
```

```
      f7=x15
      f6=x13
      f5=x11
      f4=x9
      f3=x7
      f2=x5
      f1=x3
      f0=1.d0
      go to 463
  462 t=w*pqsq
      if (t-27.9d0)4620,4621,4621
 4620 x=10.d0*(t+0.05d0)
      it=x
      ti=it
      it=it+1
      delt=t-0.1d0*ti
      delt2=0.5d0*delt
      delt3=-.33333333d0*delt
c     correction here 8/70-wyh-thd got info from basch
c     delt4=.25d0*delt
      delt4=-0.25d0*delt
      tf0=str0(it)
      tf1=str1(it)
      tf2=str2(it)
      tf3=str3(it)
      tf4=str4(it)
      tf5=str5(it)
      tf6=str6(it)
      tf7=str7(it)
      tf8=str8(it)
      tf9=str9(it)
      tf10=str10(it)
      tf11=str11(it)
      tf12=str12(it)
      go to (800,801,802,803,804,805,806,807,808),mxyz
  808 f8=tf8+delt*(-tf9+delt2*(tf10+delt3*(tf11+delt4*tf12)))
  807 f7=tf7+delt*(-tf8+delt2*(tf9+delt3*(tf10+delt4*tf11)))
  806 f6=tf6+delt*(-tf7+delt2*(tf8+delt3*(tf9+delt4*tf10)))
  805 f5=tf5+delt*(-tf6+delt2*(tf7+delt3*(tf8+delt4*tf9)))
  804 f4=tf4+delt*(-tf5+delt2*(tf6+delt3*(tf7+delt4*tf8)))
  803 f3=tf3+delt*(-tf4+delt2*(tf5+delt3*(tf6+delt4*tf7)))
  802 f2=tf2+delt*(-tf3+delt2*(tf4+delt3*(tf5+delt4*tf6)))
  801 f1=tf1+delt*(-tf2+delt2*(tf3+delt3*(tf4+delt4*tf5)))
  800 f0=tf0+delt*(-tf1+delt2*(tf2+delt3*(tf3+delt4*tf4)))
      go to 463
 4621 xd=1.d0/t
      f0=.88622692d0*dsqrt(xd)
      f1=0.5d0*xd*f0
      f2=1.5d0*xd*f1
      f3=2.5d0*xd*f2
      f4=3.5d0*xd*f3
      if (mxyz.lt.6)goto463
      f5=4.5d0*xd*f4
      f6=5.5d0*xd*f5
      f7=6.5d0*xd*f6
      f8=7.5d0*xd*f7
  463 continue
      zz(1)=1.d0
      zz(2)=z
      if (mxyz-2)380,380,382
  382 do 381life=3,mxyz
      nice=life-1
 381  zz(life)=z*zz(nice)
  380 continue
```

```
      do390nx=1,mx
      do389ny=1,my
      do388nz=1,mz
      n=nx+ny+nz-2
 388  rawint=rawint+c(nx,1)*c(ny,2)*c(nz,3)*zz(n)*f(n)
 389  continue
 390  continue
      prtint=rawint
 540  continue
 355  valint(m)=valint(m)+wx*prtint
 351  continue
 352  continue
 353  continue
      return
      end
      subroutine spdfnt (eta,valint,s,nr,ntmx,ngmx,ninmax,nsavmx)
      implicit double precision(a-h,o-z)
      dimension p(3),q(3),r(3),pa(3),pb(3),qc(3),qd(3),f(13),c(13,3),
     1mhi(3),zz(13),e(7,3)
      dimension eta(ngmx,5),s(nsavmx),valint(ninmax),nr(ntmx,3)
      common/specs/icnt,jcnt,kcnt,lcnt,ityp,jtyp,ktyp,ltyp,is,js,ks,
     1ls,if,jf,kf,lf,m,i,j,k,l
      common/nmbrs/pi,piterm,pitern,acrcy,scale
      common/gamma/f0,f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,f12
      common/inc/x3,x5,x7,x9,x11,x13,x15,x17,x19,x21,x23,x25
      equivalence(f(1),f0)
      fab=1.d0
      fcd=1.d0
      abxyz=icnt-jcnt
      cdxyz=kcnt-lcnt
      abxyz=dabs(abxyz)
      cdxyz=dabs(cdxyz)
      do 353ii=is,if
      a=eta(ii,4)
      if (i-j)680,681,680
  681 jf=ii
      fab=2.d0
  680 do 352jj=js,jf
      b=eta(jj,4)
      t1=1.d0/(a+b)
      gab=0.25d0*t1
      index =max0(ii,jj)*(max0(ii,jj)-1)/2+min0(ii,jj)
      saboo=s(index)
      if (abxyz)704,708,704
  704 continue
      p(1)=(a*eta(ii,1)+b*eta(jj,1))*t1
      p(2)=(a*eta(ii,2)+b*eta(jj,2))*t1
      p(3)=(a*eta(ii,3)+b*eta(jj,3))*t1
      go to 709
  708 p(1)=eta(ii,1)
      p(2)=eta(ii,2)
      p(3)=eta(ii,3)
  709 continue
      pa(1)=p(1)-eta(ii,1)
      pa(2)=p(2)-eta(ii,2)
      pa(3)=p(3)-eta(ii,3)
      pb(1)=p(1)-eta(jj,1)
      pb(2)=p(2)-eta(jj,2)
      pb(3)=p(3)-eta(jj,3)
      do 295kt=1,3
      ni=nr(ityp,kt)+1
      nj=nr(jtyp,kt)+1
      pakt=pa(kt)
      pbkt=pb(kt)
```

```
c     correction here 8/70-wyh-thd got info from basch
      hl0=0.d0
      hl1=0.d0
      hl2=0.d0
      hl3=0.d0
      hl4=0.d0
      hl5=0.d0
      hl6=0.d0
      go to (100,101,102,113),ni
  100 go to (103,104,105,116),nj
  101 go to (106,107,108,119),nj
  102 go to (109,110,111,122),nj
  113 go to (123,124,125,126),nj
c.....00
  103 hl0=1.d0
      go to 199
c.....10
  106 hl0=pakt
      hl1=gab
      go to 199
c.....11
  107 hl2=gab*gab
      if(abxyz)150,151,150
 151  hl0=2.d0*gab
      hl1=0.d0
      goto199
  150 hl0=pakt*pbkt+2.d0*gab
      hl1=gab*(pakt+pbkt)
      go to 199
c.....20
  109 hl2=gab*gab
      if(abxyz)152,153,152
 153  hl0=2.d0*gab
      goto199
  152 hl0=pakt*pakt+2.d0*gab
      hl1=2.d0*gab*pakt
      go to 199
c.....21
  110 gg=gab*gab
      hl3=gab*gg
      if (abxyz)154,156,154
  156 hl1=6.d0*gg
      goto199
  154 hl0=pakt*pakt*pbkt+2.d0*gab*(2.d0*pakt+pbkt)
      hl1=gab*pakt*(2.d0*pbkt+pakt)+6.d0*gg
      hl2=gg*(2.d0*pakt+pbkt)
      go to 199
c.....22
  111 gg=gab*gab
      hl4=gg*gg
      if (abxyz)157,159,157
  159 hl2=12.d0*gab*gg
      hl0=12.d0*gg
      goto199
 157  ab=pakt*pbkt
      apb=(pakt+pbkt)*(pakt+pbkt)+2.d0*ab
      hl0=ab*ab+2.d0*gab*apb+12.d0*gg
      hl1=(pakt+pbkt)*(2.d0*gab*ab+12.d0*gg)
      hl2=gg*(apb+12.d0*gab)
      hl3=2.d0*gg*gab*(pakt+pbkt)
      go to 199
c.....30
  123 gg=gab*gab
      hl3=gab*gg
```

```
      if (abxyz)840,156,840
  840 pakt2=pakt*pakt
      hl2=3.d0*gg*pakt
      hl1=3.d0*gab*(pakt2+2.d0*gab)
      hl0=pakt*(pakt2+6.d0*gab)
      go to 199
c.....31
  124 gg=gab*gab
      hl4=  gg*gg
      if (abxyz)842,159,842
  842 pakt2=pakt*pakt
      pab=pakt*pbkt
      hl3=gab*gg*(3.d0*pakt+pbkt)
      hl2=3.d0*gg*(4.d0*gab+pakt*(pakt+pbkt))
      hl1=gab*(pakt2*(pakt+3.d0*pbkt)+6.d0*gab*(3.d0*pakt+pbkt))
      hl0=pab*pakt2+6.d0*gab*(pakt2+pab)+12.d0*gg
      go to 199
c.....32
  125 gg=gab*gab
      gg2=gg*gg
      hl5=gab*gg2
      if (abxyz)843,844,843
  844 hl3=20.d0*gg2
      hl1=60.d0*gab*gg
      go to 199
  843 pakt2=pakt*pakt
      pbkt2=pbkt*pbkt
      pabkt=pakt*pbkt
      hl4=gg2*(3.d0*pakt+2.d0*pbkt)
      hl3=gab*gg*(3.d0*pakt2+6.d0*pabkt+pbkt2)+20.d0*gg2
      hl2=gg*(pakt*(pakt2+6.d0*pabkt+3.d0*pbkt2)+gab*(36.d0*pakt+24.d0*
     xpbkt))
      hl1=gab*(pabkt*(2.d0*pakt2+3.d0*pabkt)+gab*((18.d0*pakt2+36.d0*pab
     xkt+     6.d0*pbkt2)+60.d0*gab))
      hl0=pakt*(pabkt*pabkt+gab*(2.d0*pakt2+12.d0*pabkt+6.d0*pbkt2))+
     x       gg*( 36.d0*pakt+24.d0*pbkt)
      go to 199
c.....33
  126 gg=gab*gab
      gg2=gg*gg
      hl6=gg*gg2
      if (abxyz)845,846,845
  846 hl4=30.d0*gab*gg2
      hl2=180.d0*gg2
      hl0=120.d0*gab*gg
      go to 199
  845 pakt2=pakt*pakt
      pbkt2=pbkt*pbkt
      pabkt=pakt*pbkt
      pab=3.d0*(pakt+pbkt)
      pkt=pakt2+3.d0*pabkt+pbkt2
      big=pakt2*(pakt+9.d0*pbkt)+pbkt2*(pbkt+9.d0*pakt)
      hl5=gab*gg2*pab
      hl4=gg2*(3.d0*pkt+30.d0*gab)
      hl3=gab*gg*(big+gab*20.d0*pab)
      hl2=gg*(3.d0*pabkt*pkt+gab*(36.d0*pkt+gab*180.d0))
      hl1=gab*(pab*pabkt*pabkt+gab*(6.d0*big+60.d0*gab*pab))
      hl0=pabkt*pabkt*pabkt+gab*(6.d0*pabkt*pkt+gab*(36.d0*pkt+120.d0*ga
     xb))
      go to 199
c.....01
  104 hl0=pbkt
      hl1=gab
      go to 199
```

```
c.....02
  105 hl2=gab*gab
      if (abxyz)160,153,160
  160 hl0=pbkt*pbkt+2.d0*gab
      hl1=pbkt*2.d0*gab
      go to 199
c......03
  116 temp=pakt
      pakt=pbkt
      pbkt=temp
      go to 123
c.....12
  108 gg=gab*gab
      hl3=gab*gg
      if (abxyz)162,156,162
  162 hl0=pbkt*pbkt*pakt+2.d0*gab*(2.d0*pbkt+pakt)
      hl1=gab*pbkt*(2.d0*pakt+pbkt)+6.d0*gg
      hl2=gg*(2.d0*pbkt+pakt)
      go to 199
c......13
  119 temp=pakt
      pakt=pbkt
      pbkt=temp
      go to 124
c......23
  122 temp=pakt
      pakt=pbkt
      pbkt=temp
      go to 125
  199 continue
      e(1,kt)=hl0
      e(2,kt)=hl1
      e(3,kt)=hl2
      e(4,kt)=hl3
      e(5,kt)=hl4
      e(6,kt)=hl5
      e(7,kt)=hl6
  295 continue
      if(ii-jj)431,432,431
  432 fab=1.d0
  431 do 351kk=ks,kf
      cx=eta(kk,4)
      if (k-1)683,682,683
  682 lf=kk
      fcd=2.d0
 683  do355ll=ls,lf
      if (kk-ll)436,435,436
  435 fcd=1.d0
  436 continue
      d=eta(ll,4)
      t2=1.d0/(cx+d)
      gcd=0.25d0*t2
      z=1.d0/(gab+gcd)
      index =max0(kk,ll)*(max0(kk,ll)-1)/2+min0(kk,ll)
      scdoo=s(index)
      w=0.25d0*z
      wx=piterm*dsqrt(w)*saboo*scdoo*fab*fcd
      test=2.d0*wx
      prtint=0.d0
      if (dabs(test)-acrcy)540,750,750
  750 if (cdxyz)705,706,705
  705 continue
      q(1)=(cx*eta(kk,1)+d*eta(ll,1))*t2
      q(2)=(cx*eta(kk,2)+d*eta(ll,2))*t2
```

```
      q(3)=(cx*eta(kk,3)+d*eta(ll,3))*t2
      go to 707
  706 q(1)=eta(kk,1)
      q(2)=eta(kk,2)
      q(3)=eta(kk,3)
  707 r(1)=p(1)-q(1)
      r(2)=p(2)-q(2)
      r(3)=p(3)-q(3)
      pqsq=r(1)*r(1)+r(2)*r(2)+r(3)*r(3)
      qc(1)=q(1)-eta(kk,1)
      qc(2)=q(2)-eta(kk,2)
      qc(3)=q(3)-eta(kk,3)
      qd(1)=q(1)-eta(ll,1)
      qd(2)=q(2)-eta(ll,2)
      qd(3)=q(3)-eta(ll,3)
      if (pqsq-1.0d-16)461,461,462
  461 continue
      f12=x25
      f11=x23
      f10=x21
      f9=x19
      f8=x17
      f7=x15
      f6=x13
      f5=x11
      f4=x9
      f3=x7
      f2=x5
      f1=x3
      f0=1.d0
      go to 463
  462 t=w*pqsq
      y=exp(-t)
      f12=fmch(12,t,y)
      t=2.d0*t
      f11=(t*f12+y)*x23
      f10=(t*f11+y)*x21
      f9=(t*f10+y)*x19
      f8=(t*f9+y)*x17
      f7=(t*f8+y)*x15
      f6=(t*f7+y)*x13
      f5=(t*f6+y)*x11
      f4=(t*f5+y)*x9
      f3=(t*f4+y)*x7
      f2=(t*f3+y)*x5
      f1=(t*f2+y)*x3
      f0=(t*f1+y)
  463 continue
      rawint=0.d0
      do 346kt=1,3
      ni=nr(ityp,kt)
      nj=nr(jtyp,kt)
      nk=nr(ktyp,kt)
      nl=nr(ltyp,kt)
      msum=ni+nj+nk+nl+1
      if (msum-1)2,2,610
    2 c(1,kt)=1.d0
      go to 999
  610 qckt=qc(kt)
      qdkt=qd(kt)
      rkt=-r(kt)
      hl0=e(1,kt)
      hl1=e(2,kt)
      hl2=e(3,kt)
```

```
      hl3=e(4,kt)
      hl4=e(5,kt)
      hl5=e(6,kt)
      hl6=e(7,kt)
      nk=nk+1
      nl=nl+1
c     correction here 8/70-wyh-thd got info from basch
      hm0=0.d0
      hm1=0.d0
      hm2=0.d0
      hm3=0.d0
      hm4=0.d0
      hm5=0.d0
      hm6=0.d0
      go to (200,201,202,213),nk
  200 go to (203,204,205,216),nl
  201 go to (206,207,208,219),nl
  202 go to (209,210,211,222),nl
  213 go to (223,224,225,226),nl
c.....00
  203 hm0=1.d0
      goto299
c.....10
  206 hm0=qckt
      hm1=-gcd
      goto299
c.....11
  207 hm2=gcd*gcd
      if(cdxyz)250,251,250
 251  hm0=2.d0*gcd
      goto299
  250 hm0=qckt*qdkt+2.d0*gcd
      hm1=-gcd*(qckt+qdkt)
      goto299
c.....20
  209 hm2=gcd*gcd
      if(cdxyz)252,253,252
 253  hm0=2.d0*gcd
      goto299
  252 hm0=qckt*qckt+2.d0*gcd
      hm1=-2.d0*gcd*qckt
      goto299
c.....21
  210 gg=gcd*gcd
      hm3=-gcd*gg
      if(cdxyz)254,256,254
  256 hm1=-6.d0*gg
      goto299
  254 hm0=qckt*qckt*qdkt+2.d0*gcd*(2.d0*qckt+qdkt)
      hm1=-(gcd*qckt*(2.d0*qdkt+qckt)+6.d0*gg)
      hm2=gg*(2.d0*qckt+qdkt)
      goto299
c.....22
  211 gg=gcd*gcd
      hm4=gg*gg
      if(cdxyz)257,259,257
  259 hm2=12.d0*gcd*gg
      hm0=12.d0*gg
      goto299
  257 cd=qckt*qdkt
      cpd=(qckt+qdkt)*(qckt+qdkt)+2.d0*cd
      hm0=cd*cd+2.d0*gcd*cpd+12.d0*gg
      hm1=-(qckt+qdkt)*(2.d0*gcd*cd+12.d0*gg)
      hm2=gg*(cpd+12.d0*gcd)
```

```
      hm3=-2.d0*gg*gcd*(qckt+qdkt)
      goto299
c......30
  223 gg=gcd*gcd
      hm3=-gcd*gg
      if (cdxyz)280,256,280
  280 qckt2=qckt*qckt
      hm2=3.d0*gg*qckt
      hm1=-3.d0*gcd*(qckt2+2.d0*gcd)
      hm0=qckt*(qckt2+6.d0*gcd)
      go to 299
c......31
  224 gg=gcd*gcd
      hm4=gg*gg
      if (cdxyz)282,259,282
  282 qckt2=qckt*qckt
      qcd=qckt*qdkt
      hm3=-gcd*gg*(3.d0*qckt+qdkt)
      hm2=3.d0*gg*(4.d0*gcd+qckt*(qckt+qdkt))
      hm1=-(gcd*(qckt2*(qckt+3.d0*qdkt)+6.d0*gcd*(3.d0*qckt+qdkt)))
      hm0=qcd*qckt2+6.d0*gcd*(qckt2+qcd)+12.d0*gg
      go to 299
c......32
  225 gg=gcd*gcd
      gg2=gg*gg
      hm5=-gcd*gg2
      if (cdxyz)283,284,283
  284 hm3=-20.d0*gg2
      hm1=-60.d0*gcd*gg
      go to 299
  283 qckt2=qckt*qckt
      qdkt2=qdkt*qdkt
      qcdkt=qckt*qdkt
      hm4=gg2*(3.d0*qckt+2.d0*qdkt)
      hm3=-(gcd*gg*(3.d0*qckt2+6.d0*qcdkt+qdkt2)+20.d0*gg2)
      hm2=gg*(qckt*(qckt2+6.d0*qcdkt+3.d0*qdkt2)+gcd*(36.d0*qckt+24.d0*
     xqdkt))
      hm1=-(gcd*(qcdkt*(2.d0*qckt2+3.d0*qcdkt)+gcd*((18.d0*qckt2+36.d0*q
     xcdkt    +6.d0*qdkt2)+60.d0*gcd)))
      hm0=qckt*(qcdkt*qcdkt+gcd*(2.d0*qckt2+12.d0*qcdkt+6.d0*qdkt2))
     x       +gg*( 36.d0*qckt+24.d0*qdkt)
      go to 299
c......33
  226 gg=gcd*gcd
      gg2=gg*gg
      hm6=gg*gg2
      if (cdxyz)285,286,285
  286 hm4=30.d0*gcd*gg2
      hm2=180.d0*gg2
      hm0=120.d0*gcd*gg
      go to 299
  285 qckt2=qckt*qckt
      qdkt2=qdkt*qdkt
      qcdkt=qckt*qdkt
      cdq=3.d0*(qckt+qdkt)
      qkt=qckt2+3.d0*qcdkt+qdkt2
      big=qckt2*(qckt+9.d0*qdkt)+qdkt2*(qdkt+9.d0*qckt)
      hm5=-gcd*gg2*cdq
      hm4=gg2*(3.d0*qkt+30.d0*gcd)
      hm3=-(gcd*gg*(big+gcd*20.d0*cdq))
      hm2=gg*(3.d0*qcdkt*qkt+gcd*(36.d0*qkt+gcd*180.d0))
      hm1=-(gcd*(cdq*qcdkt*qcdkt+gcd*(6.d0*big+60.d0*gcd*cdq)))
      hm0=qcdkt*qcdkt*qcdkt+gcd*(6.d0*qcdkt*qkt+gcd*(36.d0*qkt+120.d0*gc
     xd))
```

```
      go to 299
c.....01
  204 hm0=qdkt
      hm1=-gcd
      goto299
c.....02
  205 hm2=gcd*gcd
      if (cdxyz)260,253,260
  260 hm0=qdkt*qdkt+2.d0*gcd
      hm1=-2.d0*gcd*qdkt
      goto299
c.....03
  216 temp=qckt
      qckt=qdkt
      qdkt=temp
      go to 223
c.....12
  208 gg=gcd*gcd
      hm3=-gcd*gg
      if (cdxyz)262,256,262
  262 hm0=qdkt*qdkt*qckt+2.d0*gcd*(2.d0*qdkt+qckt)
      hm1=-(gcd*qdkt*(2.d0*qckt+qdkt)+6.d0*gg)
      hm2=gg*(2.d0*qdkt+qckt)
      go to 299
c......13
  219 temp=qckt
      qckt=qdkt
      qdkt=temp
      go to 224
c......23
  222 temp=qckt
      qckt=qdkt
      qdkt=temp
      go to 225
  299 continue
      hlm0=hl0*hm0
      c(1,kt)=hlm0
      hlm1=hl0*hm1+hl1*hm0
      c(2,kt)=rkt*hlm1
      if (msum-2)999,999,802
  802 continue
      hlm2=hl0*hm2+hl1*hm1+hl2*hm0
      c(2,kt)=c(2,kt)-2.d0*hlm2
      rkt2=rkt*rkt
      c(3,kt)=rkt2*hlm2
      if (msum-3)999,999,803
  803 continue
      hlm3=hl0*hm3+hl1*hm2+hl2*hm1+hl3*hm0
      c(3,kt)=c(3,kt)-6.d0*rkt*hlm3
      rkt3=rkt*rkt2
      c(4,kt)=rkt3*hlm3
      if (msum-4)999,999,804
  804 continue
      hlm4=hl0*hm4+hl1*hm3+hl2*hm2+hl3*hm1+hl4*hm0
      c(3,kt)=c(3,kt)+12.d0*hlm4
      c(4,kt)=c(4,kt)-12.d0*rkt2*hlm4
      rkt4=rkt*rkt3
      c(5,kt)=rkt4*hlm4
      if (msum-5)999,999,805
  805 continue
      hlm5=hl5*hm0+hl4*hm1+hl3*hm2+hl2*hm3+hl1*hm4+hl0*hm5
      c(4,kt)=c(4,kt)+60.d0*rkt*hlm5
      c(5,kt)=c(5,kt)-20.d0*rkt3*hlm5
      rkt5=rkt*rkt4
```

```
      c(6,kt)=rkt5*hlm5
      if (msum-6)999,999,806
  806 continue
      hlm6=hl6*hm0+hl5*hm1+hl4*hm2+hl3*hm3+hl2*hm4+hl1*hm5+hl0*hm6
      c(4,kt)=c(4,kt)-120.d0*hlm6
      c(5,kt)=c(5,kt)+180.d0*rkt2*hlm6
      c(6,kt)=c(6,kt)-30.d0*rkt4*hlm6
      rkt6=rkt*rkt5
      c(7,kt)=rkt6*hlm6
      if (msum-7)999,999,807
  807 continue
      hlm7=hl1*hm6+hl2*hm5+hl3*hm4+hl4*hm3+hl5*hm2+hl6*hm1
      c(5,kt)=c(5,kt)-840.d0*rkt*hlm7
      c(6,kt)=c(6,kt)+420.d0*rkt3*hlm7
      c(7,kt)=c(7,kt)-42.d0*rkt5*hlm7
      rkt7=rkt*rkt6
      c(8,kt)=rkt7*hlm7
      if (msum-8)999,999,808
  808 continue
      hlm8=hl2*hm6+hl3*hm5+hl4*hm4+hl5*hm3+hl6*hm2
      c(5,kt)=c(5,kt)+1680.d0*hlm8
      c(6,kt)=c(6,kt)-3360.d0*rkt2*hlm8
      c(7,kt)=c(7,kt)+840.d0*rkt4*hlm8
      c(8,kt)=c(8,kt)-56.d0*rkt6*hlm8
      rkt8=rkt*rkt7
      c(9,kt)=rkt8*hlm8
      if (msum-9)999,999,809
  809 continue
      hlm9=hl3*hm6+hl4*hm5+hl5*hm4+hl6*hm3
      c(6,kt)=c(6,kt)+15120.d0*rkt*hlm9
      c(7,kt)=c(7,kt)-10080.d0*rkt3*hlm9
      c(8,kt)=c(8,kt)+1512.d0*rkt5*hlm9
      c(9,kt)=c(9,kt)-72.d0*rkt7*hlm9
      rkt9=rkt*rkt8
      c(10,kt)=rkt9*hlm9
      if (msum-10)999,999,810
  810 continue
      hlm10=hl4*hm6+hl5*hm5+hl6*hm4
      c(6,kt)=c(6,kt)-30240.d0*hlm10
      c(7,kt)=c(7,kt)+75600.d0*rkt2*hlm10
      c(8,kt)=c(8,kt)-25200.d0*rkt4*hlm10
      c(9,kt)=c(9,kt)+2520.d0*rkt6*hlm10
      c(10,kt)=c(10,kt)-90.d0*rkt8*hlm10
      rkt10=rkt*rkt9
      c(11,kt)=rkt10*hlm10
      if (msum-11)999,999,811
  811 continue
      hlm11=hl5*hm6+hl6*hm5
      c(7,kt)=c(7,kt)-332640.d0*rkt*hlm11
      c(8,kt)=c(8,kt)+277200.d0*rkt3*hlm11
      c(9,kt)=c(9,kt)-55440.d0*rkt5*hlm11
      c(10,kt)=c(10,kt)+3960.d0*rkt7*hlm11
      c(11,kt)=c(11,kt)-110.d0*rkt9*hlm11
      rkt11=rkt*rkt10
      c(12,kt)=rkt11*hlm11
      if (msum-12)999,999,812
  812 continue
      hlm12=hl6*hm6
      c(7,kt)=c(7,kt)+665280.d0*hlm12
      c(8,kt)=c(8,kt)-1995840.d0*rkt2*hlm12
      c(9,kt)=c(9,kt)+831600.d0*rkt4*hlm12
      c(10,kt)=c(10,kt)-110880.d0*rkt6*hlm12
      c(11,kt)=c(11,kt)+5940.d0*rkt8*hlm12
      c(12,kt)=c(12,kt)-132.d0*rkt10*hlm12
```

```
      rkt12=rkt*rkt11
      c(13,kt)=rkt12*hlm12
  999 continue
      mhi(kt)=msum
 346  continue
      mx=mhi(1)
      my=mhi(2)
      mz=mhi(3)
      mxyz=mx+my+mz-2
      zz(1)=1.d0
      zz(2)=z
      if (mxyz-2)380,380,382
  382 do 381life=3,mxyz
      nice=life-1
 381  zz(life)=z*zz(nice)
  380 continue
      do390nx=1,mx
      do389ny=1,my
      do388nz=1,mz
      n=nx+ny+nz-2
 388  rawint=rawint+c(nx,1)*c(ny,2)*c(nz,3)*zz(n)*f(n)
 389  continue
 390  continue
      prtint=rawint
 540  continue
 355  valint(m)=valint(m)+wx*prtint
 351  continue
 352  continue
 353  continue
      return
      end
      subroutine gfunct (l,m,a,b,p,t,g,n)
      implicit double precision(a-h,o-z)
      dimension g(7,3)
      ll=l+1
      mm=m+1
      go to (100,101,102,103),ll
  100 go to (110,111,112,113),mm
  101 go to (120,121,122,123),mm
  102 go to (130,131,132,133),mm
  103 go to (140,141,142,143),mm
c......00
  110 g(1,n)=1.d0
      go to 300
c......01
  111 g(1,n)=b
      g(2,n)=-p
      go to 300
c......02
  112 g(1,n)=b*b+0.5d0*t
      g(2,n)=-2.d0*b*p-0.5d0*t
      g(3,n)=p*p
      go to 300
c......03
  113 temp=a
      a=b
      b=temp
      go to 140
c......10
  120 g(1,n)=a
      g(2,n)=-p
      go to 300
c......11
  121 g(1,n)=a*b+0.5d0*t
```

```
      g(2,n)=-p*(a+b)-0.5d0*t
      g(3,n)=p*p
      go to 300
c......12
  122 g(1,n)=b*b*a+0.5d0*t*(a+2.d0*b)
      g(2,n)=-p*b*(2.d0*a+b)-0.5d0*t*((a+2.d0*b)+3.d0*p)
      g(3,n)=p*(p*(a+2.d0*b)+1.5d0*t)
      g(4,n)=-p*p*p
      go to 300
c......13
  123 temp=a
      a=b
      b=temp
      go to 141
c......20
  130 g(1,n)=a*a+0.5d0*t
      g(2,n)=-2.d0*a*p-0.5d0*t
      g(3,n)=p*p
      go to 300
c......21
  131 g(1,n)=a*a*b+0.5d0*t*(2.d0*a+b)
      g(2,n)=-p*a*(a+2.d0*b)-0.5d0*t*((2.d0*a+b)+3.d0*p)
      g(3,n)=p*(p*(2.d0*a+b)+1.5d0*t)
      g(4,n)=-p*p*p
      go to 300
c......22
  132 aa=a*a
      bb=b*b
      pp=p*p
      ab=4.d0*a*b
      g(1,n)=aa*bb+t*(0.5d0*(aa+ab+bb)+0.75d0*t)
      g(2,n)=-(2.d0*p*(aa*b+a*bb)+t*(0.5d0*(aa+ab+bb)+3.d0*(a+b)*p+1.5
     xd0*t))
      g(3,n)=pp*((aa+ab+bb)+3.d0*t)+t*(3.d0*p*(a+b)+0.75d0*t)
      g(4,n)=-(pp*(2.d0*p*(a+b)+3.d0*t))
      g(5,n)=pp*pp
      go to 300
c......23
  133 temp=a
      a=b
      b=temp
      go to 142
c......30
  140 g(1,n)=a*(a*a+1.5d0*t)
      g(2,n)=-3.d0*(a*(a*p+0.5d0*t)+0.5d0*p*t)
      g(3,n)=3.d0*p*(a*p+0.5d0*t)
      g(4,n)=-p*p*p
      go to 300
c......31
  141 p2=p*p
      t2=t*t
      a2=a*a
      ab=a*b
      f0=a2*ab
      f1=a2*(a+3.d0*b)
      f2=3.d0*a*(a+b)
      f3=3.d0*a+b
      g(1,n)=f0+.5d0*t*f2+.75d0*t2
      g(2,n)=-(p*f1+.5d0*t*f2+1.5d0*p*t*f3+1.5d0*t2)
      g(3,n)=p2*f2+1.5d0*p*t*f3+3.d0*t*p2+.75d0*t2
      g(4,n)=-(p*p2*f3+3.d0*t*p2)
      g(5,n)=p2*p2
      go to 300
c......32
```

```
  142 p2=p*p
      a2=a*a
      b2=b*b
      ab=a*b
      t2=t*t
      a3=3.d0*a2+6.d0*ab+b2
      a1=a2+6.d0*ab+3.d0*b2
      g(1,n)=a*a2*b2+0.5d0*t*a1*a+0.75d0*t2*(3.d0*a+2.d0*b)
      g(2,n)=-(ab*p*(2.d0*a2+3.d0*ab)+0.5d0*t*a*a1+1.5d0*t*a3*p+1.5d0*t2
     x*(3.d0*a+2.d0*b)+3.75d0*p*t2)
      g(3,n)=p2*a*a1+1.5d0*p*t*a3+3.d0*(3.d0*a+2.d0*b)*(t*p2+.25d0*t2)+7
     x.5d0*p*t2
      g(4,n)=-(p2*(p*a3+t*(9.d0*a+6.d0*b)+5.d0*p*t)+3.75d0*p*t2)
      g(5,n)=p*p2*(p*(3.d0*a+2.d0*b)+5.d0*t)
      g(6,n)=-p*p2*p2
      go to 300
c......33
  143 p2=p*p
      a2=a*a
      b2=b*b
      t2=t*t
      ab=a*b
      f0=a2*b2*ab
      f1=3.d0*a2*b2*(a+b)
      f2=3.d0*ab*(a2+3.d0*ab+b2)
      f3=a2*(a+9.d0*b)+b2*(b+9.d0*a)
      f4=3.d0*(a2+3.d0*ab+b2)
      f5=3.d0*(a+b)
      g(1,n)=f0+.5d0*t*f2+.75d0*t2*f4+1.875d0*t*t2
      g(2,n)=-(p*f1+.5d0*t*f2+1.5d0*t*p*f3+1.5d0*t2*f4+3.75d0*p*t2*f5+5.
     x625d0*t*  t2)
      g(3,n)=p2*f2+1.5d0*p*t*f3+3.d0*f4*(t*p2+.25d0*t2)+7.5d0*p*t2*f5+11
     x.25d0*t2* (.5d0*t+p2)
      g(4,n)=-(p2*(p*f3+3.d0*t*f4)+5.d0*f5*p*t*(p2+.75d0*t)+t2*(22.50d0*
     xp2+1.875d0*t))
      g(5,n)=p2*(f4*p2+5.d0*p*t*f5+7.5d0*p2*t+11.25d0*t2)
      g(6,n)=-(p2*p2*(p*f5+7.5d0*t))
      g(7,n)=p2*p2*p2
  300 continue
      return
      end
      double precision function tanh3(x)
      implicit double precision(a-h,o-z)
c     tanh(x)=x*(1.0+tanh3(x))
      z=x*x
      tanh3=-z*0.3333333333333333d0*(1.0d0-z*0.4d0*(1.0d0-
     # z*0.4047619047619048d0*(1.0d0-z*0.4052287581699345d0*(1.0d0-
     # z*0.4052785923753667d0*(1.0d0-z*0.4052840550669783d0*(1.0d0-
     # z*0.4052846591850437d0*(1.0d0-z*0.4052847261979033d0*(1.0d0-
     # z*0.4052847336393726d0*(1.0d0-z*0.4052847344660274d0*(1.0d0-
     # z*0.4052847345578709d0)))))))))
      return
      end
      double precision function fa(n)
c
      implicit double precision(a-h,o-z)
c     n.le.11
      dimension gammo(6),gamme(6)
      common/aaacom/zeta
      data gammo/0.5d0,0.5d0,1.d0,3.d0,12.d0,60.d0/,gamme/0.5d0,0.25d0,
     x0.375d0,0.9375d0,3.28125d0,14.765625d0/,sqrpi/1.77245385090d0/
c
      if (mod(n,2).eq.0)goto10
c
```

```
      k=(n-1)/2
      fa=gammo(k+1)/zeta**(k+1)
      return
   10 k=n/2
      gms=gamme(k+1)*sqrpi
      fa=gms/(zeta**k*dsqrt(zeta))
      return
       end
      subroutine vvmval(itype,n,x,vm,v)
c    version #1 may  9, 1985 c.woodward
      implicit double precision(a-h,o-z)
      parameter(syslim=88.0d0)
      common/erfp/dk,xij
      data srpi/1.77245385090d0/
      if (itype.gt.1)goto20
      go to (1,2,3,4,5,6),n
      if(n.ne.0)goto100
c
c     n=0
      call dawv(daws,daws3,x)
      vm=2.0d0*daws*srpi
      if(itype.eq.0)return
      if(x.lt.0.3d0)goto11
      v=     srpi-vm/(2.0d0*x)
      return
   11 v=-srpi*daws3
      return
c
c     n=1
    1 call errv(errf,errf3,x,ex2,z)
      vm=2.0d0*errf
      if(itype.eq.0)return
      if(x.lt.0.3d0)goto21
      x2=x*x
      v    =(x-0.5d0/x)*vm+exp(-x2)
      return
   21 v=ex2*(z+(z-1.0d0)*errf3)
      return
c
c     n=2
    2 vm   =srpi*x
      if(itype.eq.0)return
      v    =x*vm
      return
c
c     n=3
    3 call errv(errf,errf3,x,ex2,z)
      vm1=2.0d0*errf
      if(x.lt.0.3d0)goto31
      x2=x*x
      v2=(x-0.5d0/x)*vm1+exp(-x2)
      go to 32
   31 v2=ex2*(z+(z-1.0d0)*errf3)
   32 continue
      vm=  x*v2+vm1
      v=(x2+0.5d0)*v2+x*vm1
      return
c
c     n=4
    4 x2=x*x
      vm   =srpi*x*(x2+1.5d0)
      if(itype.eq.0)return
      v    =srpi*x2*(x2+2.5d0)
      return
```

```
c
c     n=5
    5 call errv(errf,errf3,x,ex2,z)
      x2=x*x
      if (x.lt.0.3d0)goto42
      vm=(0.75d0+x2*(3.0d0+x2))*2.0d0*errf+x*(x2+2.5d0)*exp(-x2)
      v=(-0.375d0+x2*(2.25+x2*(4.5d0+x2)))*2.0d0*errf/x+
     #   (0.75d0+x2*(4.0d0+x2))*exp(-x2)
      return
   42 vm=2.0d0*x*exp(-x2)*((0.75d0+x2*(3.0d0+x2))*errf3+
     #    2.0d0+x2*(3.5d0+x2))
      v=2.0d0*exp(-x2)*((-0.375d0+x2*(2.25d0+x2*(4.5d0+x2)))*
     #  errf3+x2*(4.25d0+x2*(5.0d0+x2)))
      return
c
c   n=6
    6 x2=x*x
      vm=srpi*x*(3.75d0+x2*(5.0d0+x2))
      if (itype.eq.0)return
      v=srpi*x2*(8.75d0+x2*(7.0d0+x2))
      return
c
c     integrals involving the error function
c
   20 srxij=dsqrt(xij)
      dks=dk*dk
      x2=x*x
      exa=0.0d0
      if(x2 .lt.syslim)exa=exp(-x2)
      call errv(errf,errf3,x,ex2,z)
c
      go to (100,120,100,130),n-2
c
c n=2
c
      v=errf/(2.0d0*dk*srxij)
      vm=(exa*x*xij+errf*(2.0d0*dks-xij))/
     #    (4.0d0*dks*xij*srxij)
      return
c
c  n=4
c
  120 xijs=xij*xij
      v=(xij*exa*x*(2.0d0-xij*x*x/dks)+
     #    errf*(2.0d0*dks+xij))/
     #    (4.0d0*dk*xijs*srxij)
      vm=(exa*xij*x*(6.0d0*dks+xij+
     #     x*x*xij*(2.0d0*xij*x*x/dks-6.0d0))+
     #     errf*(4.0d0*dks*(dks+xij)
     #     -xijs))/(8.0d0*xijs*xij*srxij*dks)
      return
c
c  n=6
c
  130 v=(exa*x*xij*(8.0d0*dks+12.0d0*xij
     #    +x*x*xij*(-12.0d0-10.0d0*xij/dks+
     #    x*x*xij*(8.0d0+xij*(3.0d0-2.0d0*x*x)/dks)/dks))+
     #    errf*(3.0d0*xij*xij+12.0d0*xij*dk*dk+
     #    4.0d0*dk**4.0d0))/(8.0d0*xij*xij*xij*xij*srxij*dk)
      return
  100 write(60,101)itype,n
  101 format(1h1,' *** itype =',i3,'    n =',i3,' for vval')
      stop
      end
```

```
      subroutine dawv(daws,daws3,xx)
      implicit double precision(a-h,o-z)
      common/dawson/da(1000)
      common/dawsf/a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15
     # ,a16,a17
      if(xx)1,30,2
    1 x=-xx
      s=-1.0d0
      go to 3
    2 x=xx
      s=1.0d0
    3 if(x.lt.0.3d0)goto40
      if(x-9.995d0)10,20,20
c     x less than 9.995
   10 a=100.0d0*x
      j=a+0.5d0
      l=j+1
      b100=j
      b=.01d0*b100
      d=x-b
      d2=d*2.0d0
      c0=da(l)
      c1=d*(1.0d0-2.0d0*b*c0)
      c2=-d2*(b*c1+d*c0)*0.5d0
      c3=-d2*(b*c2+d*c1)*0.3333333333333d0
      c4=-d2*(b*c3+d*c2)*0.25d0
      c5=-d2*(b*c4+d*c3)*0.2d0
      daws=(c0+c1+c2+c3+c4+c5)*s
      return
c     x greater than 9.995
   20 xi=0.5d0/x
      xi2=xi*xi
      daws=xi*(1.0d0+xi2*(2.0d0+xi2*(12.0d0+xi2*(120.0d0+xi2*
     #  1680.0d0))))*s
      return
   30 daws=0.0d0
      daws3=0.0d0
      return
   40 z=2.0d0*x*x
      daws3=-z*a3*(1.0d0-z*a5*(1.0d0-z*a7*(1.0d0-z*a9
     # *(1.0d0-z*a11*(1.0d0-z*a13*(1.0d0-z*a15*(1.0d0-z*a17)))))))
      daws=xx*(1.0d0+daws3)
      return
      end
      subroutine errv(errf,errf3,xx,ex2,z)
      implicit double precision(a-h,o-z)
      common/errfun/err(550)
      common/dawsf/a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15
     # ,a16,a17
      if(xx)1,40,2
    1 x=-xx
      s=-1.0d0
      go to 3
    2 x=xx
      s=1.0d0
    3 if(x.lt.0.3d0)goto30
      if(x.gt.5.495d0)goto20
   10 a=100.0d0*x
      j=a+0.5d0
      l=j+1
      b100=j
      b=.01d0*b100
      d=x-b
      b2=b*2.0d0
```

```
      exb=exp(-b*b)
      y0=err(1)
      y1=exb
      y2=-b2*y1
      y3=-b2*y2-2.0d0*y1
      y4=-b2*y3-4.0d0*y2
      errf=(y0+d*(y1+a2*d*(y2+a3*d*(y3+a4*d*y4))))*s
      return
   20 errf=0.886226925453d0*s
      return
   30 z=2.0d0*x*x
      ex2=exp(-x*x)
      errf3=z*a3*(1.0d0+z*a5*(1.0d0+z*a7*(1.0d0+z*a9
     # *(1.0d0+z*a11*(1.0d0+z*a13*(1.0d0+z*a15*(1.0d0+z*a17)))))))
      errf=ex2*xx*(1.0d0+errf3)
      return
   40 errf=0.0d0
      z=0.0d0
      errf3=0.0d0
      ex2=0.0d0
      return
      end
c
      subroutine locpot(noc,lpskip,ncmx)
c       this subroutine reads the local potential data into three
c       arrays [nlp,clp,zlp], it also records which sites use ef-
c       fective potentials while keeping track of duplications[lpskip].
c       in the arrays nlp,clp,zlp each angular momentum potential
c       is sandwiched between two core potentials of the same type.
c       this is done to facilitate number crunching, notice this also
c       makes it neccesary to label the index of nlp,ect with several
c       indices...thus we have lstp1,lstp2,...see below.
c
c            lstp4 ldsp3  lstp3 ldsp2  lstp2 ldsp1 lspt1
c        !core ! l=3 ! core ! l=2 ! core ! l=1 ! core ! ...next input..
c      lstr4 ldsr3        ldsr2        ldsr1
c            lstr3        lstr3        lstr1
c
      implicit double precision(a-h,o-z)
      dimension lpskip(ncmx)
      character*8tlp,tlopot,void,title
      common/lptyp/tlopot(1024)
      common/int2/nlp(200),clp(200),zlp(200),lstr1(20),lstr2(20),
     1 lstr3(20),lstr4(20),lstp1(20),lstp2(20),lstp3(20),lstp4(20),
     1 ldsr1(20),ldsr2(20),ldsr3(20),ldsp1(20),ldsp2(20),ldsp3(20),
     1 dsmx(20),dpmx(20),ddmx(20),xmax(20),lmax(20)
      data void/'        '/
      lsp=0
      xmx=0.0d0
      n=0
      do 900k=1,noc
      tlp=tlopot(k)
      lpskip(k)=0
      if(tlp.eq.void)goto900
      n=n+1
      if(n.gt.20)goto998
      lpskip(k)=n
      if(k.eq.1)goto150
      k1=k-1
      do 120kk=1,k1
      if(tlp.eq.tlopot(kk))goto130
  120 continue
      go to 150
  130 lpskip(k)=lpskip(kk)
```

```
      n=n-1
      go to 900
  150 continue
      ilsp=lsp
      nbf4=0
      nbf3=0
      nbf2=0
      read(5,1004,end=999)title
      if(title.eq.void)goto999
      read(5,1007)lmx,dsmax,dpmax,ddmax
      if(dsmax.eq.0.0d0)dsmax=210.0d0
      if(dpmax.eq.0.0d0)dpmax=9000.0d0
      if(ddmax.eq.0.0d0)ddmax=12000.0d0
      write(60,1005)title
      write(60,1008)lmx,dsmax,dpmax,ddmax
      ddmx(n)=ddmax
      dsmx(n)=dsmax
      dpmx(n)=dpmax
      xmax(n)=xmx
      lmax(n)=lmx
      read(5,1004)title
      read(5,1000)nbf4
      lst=lsp+1
      lsp=lsp+nbf4
      read(5,1001)(nlp(j),zlp(j),clp(j),j=lst,lsp)
      write(60,1010)title
      write(60,1003)(nlp(j),zlp(j),clp(j),j=lst,lsp)
      lstr4(n)=lst
      lstp4(n)=lsp
      go to(230,215,200),lmx
  200 read(5,1004)title
      read(5,1000)nbf3
      lst=lsp+1
      lsp=lsp+nbf3
      read(5,1001)(nlp(j),zlp(j),clp(j),j=lst,lsp)
      write(60,1010)title
      write(60,1003)(nlp(j),zlp(j),clp(j),j=lst,lsp)
      lstr3(n)=lst
      ldsr3(n)=lst
      lstp3(n)=lsp+nbf4
      ldsp3(n)=lsp
      do 217jj=1,nbf4
      j=lsp+jj
      nlp(j)=nlp(ilsp+jj)
      zlp(j)=zlp(ilsp+jj)
      clp(j)=clp(ilsp+jj)
 217  continue
      lsp=lsp+nbf4
215   read(5,1004)title
      read(5,1000)nbf2
      lst=lsp+1
      lsp=lsp+nbf2
      read(5,1001)(nlp(j),zlp(j),clp(j),j=lst,lsp)
      write(60,1010)title
      write(60,1003)(nlp(j),zlp(j),clp(j),j=lst,lsp)
      ldsr2(n)=lst
      lstr2(n)=lst
      ldsp2(n)=lsp
      lstp2(n)=lsp+nbf4
      do 220jj=1,nbf4
      j=lsp+jj
      nlp(j)=nlp(ilsp+jj)
      zlp(j)=zlp(ilsp+jj)
      clp(j)=clp(ilsp+jj)
```

```
220   continue
      lsp=lsp+nbf4
  230 read (5,1004)title
      read(5,1000)nbfl
      lst=lsp+1
      lsp=lsp+nbfl
      read(5,1001)(nlp(j),zlp(j),clp(j),j=lst,lsp)
      write(60,1010)title
      write(60,1003)(nlp(j),zlp(j),clp(j),j=lst,lsp)
      ldsrl(n)=lst
      lstrl(n)=lst
      ldspl(n)=lsp
      lstpl(n)=lsp+nbf4
      do 240jj=1,nbf4
      j=lsp+jj
      nlp(j)=nlp(ilsp+jj)
      zlp(j)=zlp(ilsp+jj)
      clp(j)=clp(ilsp+jj)
240   continue
      lsp=lsp+nbf4
900   continue
c
      do 111m=1,n
      write(60,2000)lstrl(m),lstr2(m),lstr3(m),lstr4(m),lstpl(m),
     1 lstp2(m),lstp3(m),lstp4(m),ldsrl(m),ldsr2(m),ldsr3(m),ldspl(m),
     1 ldsp2(m),ldsp3(m),dsmx(m),dpmx(m),ddmx(m),lmax(m)
 111  continue
      do 115m=1,lsp
      write(60,2005)m,nlp(m),clp(m),zlp(m)
 115  continue
      do 117m=1,20
        write(60,2007)lpskip(m)
117   continue
2000  format(1x,i2,4x,i2,4x,i2,4x,i2,/,1x,i2,4x,i2,4x,i2,4x,i2,/,1x,i2,
     1 4x,i2,4x,
     1 i2,/,1x,i2,4x,i2,4x,i2,/,1x,f7.1,f7.1,f7.1,i4)
2005  format(2i4,3x,d15.8,4x,d15.8)
2007  format(1x,i2)
c
      return
  998 write(60,1030)
      stop
999   write(60,1020)
      stop
 1000 format(8i5)
 1001 format(i1,14x,d10.4,d19.12)
1003  format( 1x,i1,14x,d10.4,2x,d19.12)
 1004 format(8a8)
1005  format('1local potential data - ',8a8///)
 1007 format(i5,3d15.8)
1008  format(6x,'lmax =',i2,5x,'dsmax =',d15.8,5x,'dpmax =',d15.8,5x,
     1 'ddmax =',d15.8)
1010  format(//1x,8a8/)
1020  format(//10x,'**  missing  local  potential  data  **'/)
 1030 format(//10x,'**  more than 20 different local potentials **')
2400  format(1x,'enter the name of the polyin input file (file 5)')
21024  format(a16)
2600  format(1x,'enter the name of the polyin information output
     1 file (file 6)')
      end
      subroutine  vints (noc,vlist,ntype,nr,nfirst,nlast,
     $                     eta,nfmx,ncmx,ntmx,ninmax,ngmx)
      implicit double precision (a-h,o-z)
      integer*2 iil(1024),jjl(1024),itgl(1024)
```

```
      dimension ntype(nfmx),eta(ngmx,5),nfirst(nfmx),nlast(nfmx),
     1 valint(1024),nr(ntmx,3),vlist(ncmx,4)
     1,g(7,3),f(13)
      dimension lpskip(300)
      common/comfmch/pi4,ap0,ap1,ap2,ap3,ap4,ap5,ap6
      common/test/ddtest
      commonvalint
      common/gamma/f0,f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,f12
      common/inc/x3,x5,x7,x9,x11,x13,x15,x17,x19,x21,x23,x25
      common/ioind/icon(10)
      common/namtap/nitape,lstnam,notape,intnam
      common/nmbrs/pi,piterm,pitern,acrcy,scale,icanon
      common/int2/nlp(200),clp(200),zlp(200),lstr1(20),lstr2(20),
     1 lstr3(20),lstr4(20),lstp1(20),lstp2(20),lstp3(20),lstp4(20),
     1 ldsr1(20),ldsr2(20),ldsr3(20),ldsp1(20),ldsp2(20),ldsp3(20),
     1 dsmx(20),dpmx(20),ddmx(20),xmax(20),lmax(20)
      common/bfcom/xint,ityp,jtyp,a,b,nc,idum,avx,avy,avz,av,bvx,bvy,bvz
     # ,bv,pcx,pcy,pcz,pcsq,phase,charge
      equivalence(f0,f(1))
      pi4=dsqrt(pi/4.0d0)
      ddtest=0.0d0
      call locpot(noc,lpskip,ncmx)
      call dawtab
      nokk=0
      if(icon(2).ge.3)nokk=1
      if(nokk)3,3,1
1     do 2 i=1,1024
      iil(i)=0
      jjl(i)=0
2     itgl(i)=0
3     nrcnt=0
10    if(nokk)11,12,11
11    read(nitape)nints,lstrcd,iil,jjl,itgl
      go to 13
12    read(nitape)nints,lstrcd,iil,jjl,itgl
13    nrcnt=nrcnt+1
      if(nints.le.0)goto1915
      if  ( nints.le.0.or.nints.gt.ninmax)goto800
      do  916m=1,nints
      i=iil(m)
      j=jjl(m)
      itag=itgl(m)
      if (itag-1)403,402,408
  408 valint(m)=-prvint
      go to 916
  402 valint(m)=prvint
      go to 916
  403 valint(m)=0.d0
      ityp=ntype(i)
      jtyp=ntype(j)
      if(ityp.le.jtyp)goto404
      itypt=ityp
      itemp=i
      ityp=jtyp
      i=j
      jtyp=itypt
      j=itemp
  404 continue
      l1=nr(ityp,1)
      l2=nr(jtyp,1)
      m1=nr(ityp,2)
      m2=nr(jtyp,2)
      n1=nr(ityp,3)
      n2=nr(jtyp,3)
```

```
      mx=l1+l2+1
      my=m1+m2+1
      mz=n1+n2+1
      is=nfirst(i)
      if=nlast(i)
      js=nfirst(j)
      jf=nlast(j)
      do 635ii=is,if
      a=eta(ii,4)
      ax=eta(ii,1)
      ay=eta(ii,2)
      az=eta(ii,3)
      do 1635jj=js,jf
      b=eta(jj,4)
      bx=eta(jj,1)
      by=eta(jj,2)
      bz=eta(jj,3)
      t1=a+b
      t=1.d0/t1
      p1=(a*ax+b*bx)*t
      p2=(a*ay+b*by)*t
      p3=(a*az+b*bz)*t
      ab1=ax-bx
      ab2=ay-by
      ab3=az-bz
      distab=ab1*ab1+ab2*ab2+ab3*ab3
      phase=exp(-a*b*distab*t)
      soo=4.d0*pi*eta(ii,5)*eta(jj,5)
      vsoo=t*phase*0.5d0
      vlp=0.0d0
      vnai=0.d0
      do 690n=1,noc
      vx=vlist(n,1)
      vy=vlist(n,2)
      vz=vlist(n,3)
      pcx=p1-vx
      pcy=p2-vy
      pcz=p3-vz
      pcsq=pcx*pcx+pcy*pcy+pcz*pcz
      arg=t1*pcsq
      y=exp(-arg)
      f6=fmch(6,arg,y)
      arg=2.d0*arg
      if (y .lt.0.0d0)goto25
      f5=(arg*f6+y)*x11
      f4=(arg*f5+y)*x9
      f3=(arg*f4+y)*x7
      f2=(arg*f3+y)*x5
      f1=(arg*f2+y)*x3
      f0= arg*f1+y
      go to 29
 25   f5=2.0d0*ap5*x11
      f4=4.0d0*ap4*x11*x9
      f3=8.0d0*ap3*x11*x9*x7
      f2=16.0d0*ap2*x11*x9*x7*x5
      f1=32.0d0*ap1*x11*x9*x7*x5*x3
      f0=64.0d0*ap0*x11*x9*x7*x5*x3
 29   continue
      pax=p1-ax
      pbx=p1-bx
      call gfunct(l1,l2,pax,pbx,pcx,t,g,1)
      pay=p2-ay
      pby=p2-by
      call  gfunct(m1,m2,pay,pby,pcy,t,g,2)
```

160

```
      paz=p3-az
      pbz=p3-bz
      call  gfunct(n1,n2,paz,pbz,pcz,t,g,3)
      rawint=0.d0
      do 506ix=1,mx
      do 507jy=1,my
      do 508kz=1,mz
      mxyz=ix+jy+kz-2
  508 rawint=rawint+g(ix,1)*g(jy,2)*g(kz,3)*f(mxyz)
  507 continue
  506 continue
c       write(60,100) i,j,ii,jj,n,rawint
c   100 format(2x,'i=',i3,2x,'j=',i3,2x,'ii=',i3,2x,'jj=',i3,2x,
c     1 'nn=',i3,2x,'rawint=',e16.8)
      xint=0.0d0
      nc=lpskip(n)
      charge=vlist(n,4)
      dumch=charge
      if(nc)690,690,600
  600 dumch =0.0d0
      if((distab+pcsq).gt.1.0d-06)goto602
  601 call vaaa
c       write(60,110)i,j,ii,jj,n,xint,soo,charge
c   110 format(2x,'i=',i3,2x,'j=',i3,2x,'ii=',i3,2x,'jj=',i3,2x,
c     1 'nn=',i3,2x,'xint=',e16.8,2x,'vaaa',/,' soo=',f15.6,
c     1 ' charge=',f15.6)
      vlp=vlp+xint
      go to 690
  602 avx=ax-vx
      avy=ay-vy
      avz=az-vz
      avsq=avx*avx+avy*avy+avz*avz
      bvx=bx-vx
      bvy=by-vy
      bvz=bz-vz
      bvsq=bvx*bvx+bvy*bvy+bvz*bvz
      if(avsq)603,603,604
  603 bv=dsqrt(bvsq)
      call vbaa(bvx,bvy,bvz,bv,b,jtyp,a,ityp)
c       write(60,111)i,j,ii,jj,n,xint
c   111 format(2x,'i=',i3,2x,'j=',i3,2x,'ii=',i3,2x,'jj=',i3,2x,
c     1 'nn=',i3,2x,'xint=',e16.8,2x,'vbaa-1')
      vlp=vlp+xint
      go to 690
  604 if(bvsq)605,605,606
  605 av=dsqrt(avsq)
      call vbaa(avx,avy,avz,av,a,ityp,b,jtyp)
c       write(60,112)i,j,ii,jj,n,xint
c   112 format(2x,'i=',i3,2x,'j=',i3,2x,'ii=',i3,2x,'jj=',i3,2x,
c     1 'nn=',i3,2x,'xint=',e16.8,2x,'vbaa-2')
      vlp=vlp+xint
      go to 690
  606 av=dsqrt(avsq)
      bv=dsqrt(bvsq)
      call vbca
c       write(60,113)i,j,ii,jj,n,xint
    113 format(2x,'i=',i3,2x,'j=',i3,2x,'ii=',i3,2x,'jj=',i3,2x,
c     1 'nn=',i3,2x,'xint=',e16.8,2x,'vbca')
      vlp=vlp+xint
  690 vnai=vnai-rawint*dumch
c       write(60,1927),vlp
c1927  format(1x,' sum of on & off center int.',f13.6)
 1635 valint(m)=valint(m)+(vnai*vsoo+vlp)*soo
  635 continue
```

```
      prvint=valint(m)
  916 continue
 1915 if(nokk)1916,1917,1916
 1916 write(notape)nints,lstrcd,iil,jjl,itgl,valint
      go to 1918
 1917 write(notape)nints,lstrcd,iil,jjl,itgl,valint
 1918 if(lstrcd)915,10,915
  915 continue
c
c  write out results of ddmx test
c
      write(60,7114)ddtest
 7114 format(1x,' the lowest value of -ddmx that resulted in a',/,'
     1 change of the integral of more one part in 1.0e-12 is:',d15.8)
      return
  800 write (6,992)nrcnt,nints
      stop
  992 format ( / 3x, 43h**  tape  read  error  in  vints ,  nrcnt =,i5,
     x    11h ,  nints =,i10,4h  **  )
      end
      subroutine vaaa
c    this subroutine evaluates the integrals involving gaussians
c    which are centered at the location of the effective potential.
c    the effective potential reduces to two terms [vcore + vl],
c    this is due to the orthonormality of the angular part of the
c    wavefunctions.  notice that x**2, y**2, z**2 are not pure spherical
c    harmonics thus we get three terms [vcore + vs + vd] for these
c    gaussians.
c
      implicit          double precision(a-h,o-z)
      common/bfcom/xaaa,ityp,jtyp,zi,zj,kc,idum,zdum(13),zval
      common/int2/nlp(200),clp(200),zlp(200),lstrl(20),lstr2(20),
     1 lstr3(20),lstr4(20),lstp1(20),lstp2(20),lstp3(20),lstp4(20),
     1 ldsrl(20),ldsr2(20),ldsr3(20),ldsp1(20),ldsp2(20),ldsp3(20),
     1 dsmx(20),dpmx(20),ddmx(20),xmx(20),lmax(20)
      common/aaacom/zeta
c     data fourpi/12.56637060d0/   deleted
      data pi/3.14159265358979323846d0/
c
      xaaa=0.0d0
      xij=zi+zj
      if(jtyp.gt.4)goto101
      if(ityp.ne.jtyp)return
      go to (1,2,2,2),jtyp
  101 if(jtyp.gt.7)goto3
      if(ityp.eq.1)goto4
      if(ityp.le.4)return
      if(ityp.ne.jtyp)goto5
      go to 6
c
c     <s/vs/s>
c
    1 lstrt=lstr4(kc)
      lstop=lstp4(kc)
c      write(60,777)kc,lstrtl,lstopl,zval
      do 11 k=lstrt,lstop
      zeta=xij+zlp(k)
   11 xaaa=xaaa-zval*clp(k)*dsqrt(zlp(k)/zeta)/(2.0d0*xij)
c
c      write(60,780)xaaa
c 780  format(1x,'the <vcore> integral=',f15.6)
      buff=0.0d0
      lstrtl=ldsrl(kc)
      lstopl=ldspl(kc)
```

```
      do 10 k=lstrtl,lstopl
      zeta=xij+zlp(k)
      n=nlp(k)
c      write(60,778)n,zlp(k),clp(k),zi,zj,(fa(n)*clp(k))
      buff=buff+fa(n)*clp(k)
   10 xaaa=xaaa+fa(n)*clp(k)
c      write(60,2817)buff
c 2817 format(1x,' <ylm> =',d15.8)
      return
c
c     <p/vp/p>
c
    2 lstrt=lstr4(kc)
      lstop=lstp4(kc)
      do 13 k=lstrt,lstop
      zeta=xij+zlp(k)
      xaaa=xaaa-zval*clp(k)*dsqrt(zlp(k)/pi)*(fa(2)+fa(0)/xij)
     #     /xij
   13 continue
c      write(60,780)xaaa
      lstrt2=ldsr2(kc)
      lstop2=ldsp2(kc)
      do 20 k=lstrt2,lstop2
      zeta=xij+zlp(k)
      n=nlp(k)+2
   20 xaaa=xaaa+fa(n)*clp(k)
      xaaa=xaaa*0.3333333333333d0
      return
c
c     pure <d/vd/d>
c
    3 if(ityp.ne.jtyp)return
      lstrt=lstr4(kc)
      lstop=lstp4(kc)
      do 31 k=lstrt,lstop
      zeta=zlp(k)+xij
   31 xaaa=xaaa-zval*clp(k)*dsqrt(zlp(k)/pi)*(fa(4)+2.0d0*
     #     (fa(2)+fa(0)/xij)/xij)/xij
c      write(60,780)xaaa
      buff=0.0d0
      lstrt3=ldsr3(kc)
      lstop3=ldsp3(kc)
      do 30 k=lstrt3,lstop3
      zeta=xij+zlp(k)
      n=nlp(k)+4
      buff=buff+fa(n)*clp(k)
   30 xaaa=xaaa+fa(n)*clp(k)
      xaaa=xaaa/15.0d0
      buff=buff/15.0d0
c      write(60,1529)buff
c1529  format(1x,' pure d <d/vd/d>/15 =',d15.8)
      return
c
c     <s/vlm/ri*ri>
c
    4 lstrt=lstr4(kc)
      lstop=lstp4(kc)
      do 41 k=lstrt,lstop
      zeta=xij+zlp(k)
   41 xaaa=xaaa-zval*clp(k)*(3.0d0*xij+2.0d0*zlp(k))*
     # dsqrt(zlp(k)/zeta)/(4.0d0*xij*xij*zeta)
c      write(60,781)xaaa
c 781  format(1x,' <s/vcore/ri*ri>= ... xaaa=',d15.8)
      buff=0.0d0
```

```
      lstrtl=ldsrl(kc)
      lstopl=ldspl(kc)
      do 40 k=lstrtl,lstopl
      zeta=xij+zlp(k)
      n=nlp(k)+2
      buff=buff+fa(n)*clp(k)
   40 xaaa=xaaa+fa(n)*clp(k)
      xaaa=xaaa*0.3333333333333d0
      buff=buff/3.0d0
c      write(60,1427)buff
c 1427 format(1x,' <s/vs/ri*ri>/3 =',d15.8)
      return
c
c     <ri*ri/vlm/rj*rj>  i .ne. j
c
    5 xcore=0.0d0
      lstrt=lstr4(kc)
      lstop=lstp4(kc)
      do 51 k=lstrt,lstop
      zeta=zlp(k)+xij
   51 xcore=xcore-zval*clp(k)*dsqrt(zlp(k)/pi)*(fa(4)+2.0d0*
     #     (fa(2)+fa(0)/xij)/xij)/xij
      xcore=xcore/15.0d0
c      write(60,780)xcore
      lstrtl=ldsrl(kc)
      lstopl=ldspl(kc)
      lstrt3=ldsr3(kc)
      lstop3=ldsp3(kc)
      buff=0.0d0
      do 50 k=lstrtl,lstopl
      zeta=xij+zlp(k)
      n=nlp(k)+4
      buff=buff+fa(n)*clp(k)
   50 xaaa=xaaa+fa(n)*clp(k)
c      write(60,7531)buff
c 7531 format(1x,' <ri*ri/vs/rj*rj> i .ne. j=',d15.8)
      buff=0.0d0
      xddd=0.0d0
      do 55 k=lstrt3,lstop3
      zeta=xij+zlp(k)
      n=nlp(k)+4
      buff=buff+fa(n)*clp(k)
   55 xddd=xddd+fa(n)*clp(k)
c      write(60,3618)buff
c3618  format(1x,' <ri*ri/vd/rj*rj> i .ne. j=',d15.8)
      xaaa=(xaaa-0.4d0*xddd)*0.1111111111111d0
      xaaa=xaaa+xcore
      return
c
c     <ri*ri/vlm/ri*ri>
c
    6 xcore=0.0d0
      lstrt=lstr4(kc)
      lstop=lstp4(kc)
      do 61 k=lstrt,lstop
      zeta=xij+zlp(k)
   61 xcore=xcore-zval*clp(k)*dsqrt(zlp(k)/pi)*(fa(4)+2.0d0*
     #     (fa(2)+fa(0)/xij)/xij)/xij
      xcore=xcore/5.0d0
c      write(60,780)xcore
      lstrtl=ldsrl(kc)
      lstopl=ldspl(kc)
      lstrt3=ldsr3(kc)
      lstop3=ldsp3(kc)
```

```
      buff=0.0d0
      do 60 k=lstrt1,lstop1
      zeta=xij+zlp(k)
      n=nlp(k)+4
      buff=buff+fa(n)*clp(k)
   60 xaaa=xaaa+fa(n)*clp(k)
c      write(60,2615)buff
c2615  format(1x,' <ri*ri/vlm/ri*ri>=',d15.8)
      buff=0.0d0
      xddd=0.0d0
      do 65 k=lstrt3,lstop3
      zeta=xij+zlp(k)
      n=nlp(k)+4
      buff=buff+fa(n)*clp(k)
   65 xddd=xddd+fa(n)*clp(k)
c      write(60,7451)buff
c7451  format(1x,' <ri*ri/vlm/ri*ri>=',d15.8)
      xaaa=(xaaa+0.8d0*xddd)*0.1111111111111d0
      xaaa=xaaa+xcore
      return
       end
      subroutine vbaa(bax,bay,baz,ba,zeta1,jtyp,zeta2,ityp)
c
      implicit           double precision(a-h,o-z)
      common/bfcom/xbaa,idum(2),dum(2),kc,jdum,gdum(13),zval
      common/int2/nlp(200),clp(200),zlp(200),lstr1(20),lstr2(20),
     1 lstr3(20),lstr4(20),lstp1(20),lstp2(20),lstp3(20),lstp4(20),
     1 ldsr1(20),ldsr2(20),ldsr3(20),ldsp1(20),ldsp2(20),ldsp3(20),
     1 dsmx(20),dpmx(20),ddmx(20),xmax(20),lmax(20)
      common/erfp/b,xij
c     data fourpi/12.56637060d0/     deleted
      data pi/3.1415926535897932384 6d0/
      xbaa=0.0d0
      xij=zeta1+zeta2
      b=ba*zeta1
      a=b*2.0d0
      ai1=1.0d0/a
      d=b*ba
c      write(60,7251)bax,bay,baz,ba
c7251  format(1x,' vbaa=> bax=',d15.8,/,' bay=',d15.8,
c     # /,' baz=',d15.8,' ba=',d15.8)
      dx=bax/ba
      dy=bay/ba
      dz=baz/ba
      delxk=0.0d0
      delyk=0.0d0
      delzk=0.0d0
      delxl=0.0d0
      delyl=0.0d0
      delzl=0.0d0
c
      value=0.0d0
c
      go to (501,502,503,504,505,506,507,508,509,510),ityp
      go to 1
  501 go to (10,20,40,80,151,153,156,152,154,155),jtyp
      go to 1
  502 go to (21,30,50,90,171,173,176,172,174,175),jtyp
      go to 1
  503 go to (41,50,70,110,181,183,186,182,184,185),jtyp
      go to 1
  504 go to (81,90,110,150,191,193,196,192,194,195),jtyp
      go to 1
c
```

```
c  sb---sa
   10 lstrtl=lstr4(kc)
      lstopl=lstp4(kc)
c
      do 14 k=lstrtl,lstopl
      zeta=xij+zlp(k)
      exa=exp(-d*(1.0d0-zetal/xij))
      ck=-zval*clp(k)*exa
      x=zetal*ba*dsqrt(zlp(k)/(zeta*xij))
      call vvmval(2,2,x,vml,vm0)
  14  value=value+ck*vm0
c
c      write(60,2838)value
c2838  format(1x,' <s/core/s>vbaa=',d15.8)
      lstrtl=ldsrl(kc)
      lstopl=ldspl(kc)
      do 11 k=lstrtl,lstopl
      zeta=xij+zlp(k)
      n=nlp(k)
      exa=exp(-d*(1.0d0-zetal/zeta))
      srzi=1.0d0/dsqrt(zeta)
      ck=clp(k)*exa*(srzi**n)*0.5d0*ail
      x=b*srzi
      call vvmval(0,n,x,vm,v)
   11 value=value+ck*vm
c      write(60,2839)value
c 2839 format(1x,' value=',f15.9)
      go to 290
c  xb---sa
   20 c00=-bax
      c11=bax/ba
   22 if(c00.eq.0.0d0)return
c
      lstrtl=lstr4(kc)
      lstopl=lstp4(kc)
c
      bvm0=0.0d0
      bvml=0.0d0
      do 15 k=lstrtl,lstopl
      zeta=xij+zlp(k)
      exa=exp(-d*(1.0d0-zetal/xij))
      ck=-zval*clp(k)*exa
      x=zetal*ba*dsqrt(zlp(k)/(zeta*xij))
      call vvmval(2,2,x,vml,vm0)
      bvm0=bvm0+ck*vm0
      bvml=bvml+ck*vml
  15  value=value+ck*(c00*vm0+c11*vml)
c      write(60,7739)value,bvm0,bvml
c7739  format(1x,' vbaa <x/core/sa>= ',d15.8,/,
c     #         '  m0= ',d15.8,' r*ml =',d15.8)
      lstrtl=ldsrl(kc)
      lstopl=ldspl(kc)
      do 23 k=lstrtl,lstopl
      zeta=xij+zlp(k)
      n=nlp(k)
      nl=n+1
      exa=exp(-d*(1.0d0-zetal/zeta))
      srzi=1.0d0/dsqrt(zeta)
      ck=clp(k)*exa*(srzi**n)*0.5d0*ail
      x=b*srzi
      call vvmval(1,n,x,vm,v)
   23 value=value+ck*(c00*vm+c11*srzi*v)
      go to 290
c  sb---xa
```

```
   21 cll=bax/ba
   24 if(cll.eq.0.0d0)return
c
      lstrtl=lstr4(kc)
      lstopl=lstp4(kc)
c
      do 16 k=lstrtl,lstopl
      zeta=xij+zlp(k)
      exa=exp(-d*(1.0d0-zetal/xij))
      ck=-zval*clp(k)*exa
      x=zetal*ba*dsqrt(zlp(k)/(zeta*xij))
      call vvmval(2,2,x,vml,vm0)
  16  value=value+ck*vml
c
c      write(60,2271)value
c2271  format(lx,' <s/core/x>= ',d15.8)
c      value=value*cll
      lstrt2=ldsr2(kc)
      lstop2=ldsp2(kc)
      do 25 k=lstrt2,lstop2
      zeta=xij+zlp(k)
      n=nlp(k)
      nl=n+1
      exa=exp(-d*(1.0d0-zetal/zeta))
      srzi=1.0d0/dsqrt(zeta)
      ck=clp(k)*exa*(srzi**n)*0.5d0*ail
      x=b*srzi
      call vvmval(1,n,x,vm,v)
   25 value=value+ck*v*srzi
      value=cll*value
      go to 290
c  xb---xa
   30 bij=bax*bax
      c20=0.3333333333333d0
      c00=0.0d0
   31 if(bij.eq.0.0d0.and.c20.eq.0.0d0)return
      cll=-bij/ba
      c22=-cll/ba-c20
      c20=c20+c22
      cll=cll-3.0d0*c22/a
      c00s=0.0d0
c
      lstrtl=lstr4(kc)
      lstopl=lstp4(kc)
c
      bvm0=0.0d0
      bvml=0.0d0
      bvm2=0.0d0
      bvm3=0.0d0
      do 17 k=lstrtl,lstopl
      zeta=xij+zlp(k)
      exa=exp(-d*(1.0d0-zetal/xij))
      ck=-zval*clp(k)*exa
      x=zetal*ba*dsqrt(zlp(k)/(zeta*xij))
      call vvmval(2,2,x,vml,vm0)
      bvm0=bvm0+vm0*ck
      bvml=bvml+ck*vml
      cllvml=cll*vml
      call vvmval(2,4,x,vml,vm0)
      bvm2=bvm2+ck*vm0
      bvm3=bvm3+ck*vml
   17 value=value+ck*(cllvml+c20*vm0)
c
c      write(60,1719)value,bvm0,bvml,bvm2,bvm3
```

```
c1719  format(1x,' vbaa <p/core/p>= ',d15.8,/,' m0=',
c     #        d15.8,' r*m1=',d15.8,/,' r*r*m0=',d15.8,
c     #         ' r*r*r*m1=',d15.8)
      lstrt2=ldsr2(kc)
      lstop2=ldsp2(kc)
  936 do 32 k=lstrt2,lstop2
      zeta=xij+zlp(k)
      n=nlp(k)
      n1=n+1
      exa=exp(-d*(1.0d0-zeta1/zeta))
      srzi=1.0d0/dsqrt(zeta)
      ck=clp(k)*exa*(srzi**n)*0.5d0*ai1
      x=b*srzi
      call vvmval(1,n,x,vm,v)
   32 value=value+((c00+c20*n1*0.5d0/zeta)*vm+(c11+c20*x*srzi)*
     #  srzi*v    )*ck
      if(c00s.eq.0.0d0)goto290
  36  if(lmax(kc).eq.0)goto290
      lstrt=ldsr1(kc)
      lstop=ldsp1(kc)
      do 12 k=lstrt,lstop
      zeta=xij+zlp(k)
      n=nlp(k)+2
      exa=exp(-d*(1.0d0-zeta1/zeta))
      srzi=1.0d0/dsqrt(zeta)
      ck=clp(k)*exa*(srzi**n)*ai1*0.1666666666666d0
      x=b*srzi
      call vvmval(0,n,x,vm,v)
   12 value=value+ck*vm
      if (lmax(kc).lt.3)goto290
      lstrt=ldsr3(kc)
      lstop=ldsp3(kc)
      do 13 k=lstrt,lstop
      zeta=xij+zlp(k)
      n=nlp(k)+2
      exa=exp(-d*(1.0d0-zeta1/zeta))
      srzi=1.0d0/dsqrt(zeta)
      x=b*srzi
      ck=clp(k)*exa*(srzi**(n-1))*0.5d0*ai1
      c00=(d22+d20)*(n-1)*0.5d0*srzi
      c11=(d22+d20)*(a/(2.0d0*zeta)-3.0d0/a)
      call vvmval(1,n,x,vm,v)
   13 value=value+ck*(c00*vm+c11*v)
      go to 290
c  yb---sa
   40 c00=-bay
      c11=bay/ba
      go to 22
c  sb---ya
   41 c11=bay/ba
      go to 24
c  yb---xa   or   xb---ya
   50 bij=bax*bay
      c20=0.0d0
      c00=0.0d0
      go to 31
c  yb---ya
   70 bij=bay*bay
      c20=0.3333333333333d0
      c00=0.0d0
      go to 31
c  zb---sa
   80 c00=-baz
      c11=baz/ba
```

```
      go to 22
c  sb---za
   81 c11=baz/ba
      go to 24
c  xb---za   or   zb---xa
   90 bij=bax*baz
      c20=0.0d0
      c00=0.0d0
      go to 31
c  yb---za   or   zb---ya
  110 bij=bay*baz
      c20=0.0d0
      c00=0.0d0
      go to 31
c  zb---za
  150 bij=baz*baz
      c20=0.3333333333333d0
      c00=0.0d0
      go to 31
c  sa---xxb
  151 c00=bax*bax
      c20=0.3333333333333d0
      go to 34
c  sa---xyb
  152 c00=bax*bay
      c20=0.0d0
      go to 34
c  sa---yyb
  153 c00=bay*bay
      c20=0.3333333333333d0
      go to 34
c  sa---xzb
  154 c00=bax*baz
      c20=0.0d0
      go to 34
c  sa---yzb
  155 c00=bay*baz
      c20=0.0d0
      go to 34
c  sa---zzb
  156 c00=baz*baz
      c20=0.3333333333333d0
c
   34 c11=-2.0d0*c00/ba
      if(c00.eq.0.0d0.and.c20.eq.0.0d0)return
      c22=c00/(ba*ba)-c20
      c20=c20+c22
      c11=c11-3.0d0*c22/a
      c00s=0.0d0
c
      lstrt=lstr4(kc)
      lstop=lstp4(kc)
c
      bvm0=0.0d0
      bvm1=0.0d0
      bvm2=0.0d0
      bvm3=0.0d0
      do 341k=lstrt,lstop
      zeta=xij+zlp(k)
      exa=exp(-d*(1.0d0-zetal/xij))
      ck=-zval*clp(k)*exa
      x=zetal*ba*dsqrt(zlp(k)/(zeta*xij))
      call vvmval(2,2,x,vm1,vm0)
      bvm0=bvm0+ck*vm0
```

```
      bvm1=bvm1+ck*vm1
      c00vm0=c00*vm0
      c11vm1=c11*vm1
      call vvmval(2,4,x,vm1,vm0)
      bvm2=bvm2+ck*vm0
      bvm3=bvm3+ck*vm1
 341  value=value+ck*(c00vm0+c11vm1+c20*vm0)
c       write(60,1847)bvm0,bvm1,bvm2,bvm3
c 1847 format(1x,' vm0=',d15.8,2x,' r*vm1=',d15.8,/,
c     #           ' r*r*vm0=',d15.8,2x,' r*r*r*vm1=',d15.8)
      lstrt2=ldsrl(kc)
      lstop2=ldspl(kc)
      go to 936
c
c  xxa---sb
  161 bij=bax*bax
      c20=0.3333333333333d0
      k22=0.5d0
      k20=-0.1666666666666666d0
      go to 35
c  xya---sb
  162 bij=bax*bay
      c20=0.0d0
      go to 35
c  yya---sb
  163 bij=bay*bay
      c20=0.3333333333333d0
      k22=0.5d0
      k20=-0.166666666666666d0
      go to 35
c  xza---sb
  164 bij=bax*baz
      c20=0.0d0
      go to 35
c  yza---sb
  165 bij=bay*baz
      c20=0.0d0
      go to 35
c  zza---sb
  166 bij=baz*baz
      c20=0.3333333333333d0
      k22=0.0d0
      k20=0.3333333333333d0
c
   35 c00=0.0d0
      if(bij.eq.0.0d0.and.c20.eq.0.0d0)return
      c22=bij/(ba*ba)-c20
      c00s=3.0d0*c20
      c20=c20+c22
      c11=-3.0d0*c22/a
      lstrt2=lstr4(kc)
      lstop2=lstp4(kc)
c
      bvm0=0.0d0
      bvm1=0.0d0
      bvm2=0.0d0
      bvm3=0.0d0
      do 351k=lstrt2,lstop2
      zeta=xij+zlp(k)
      exa=exp(-d*(1.0d0-zetal/xij))
      ck=-zval*clp(k)*exa
      x=zetal*ba*dsqrt(zlp(k)/(zeta*xij))
      call vvmval(2,2,x,vm1,vm0)
      bvm0=bvm0+ck*vm0
```

```
      bvm1=bvm1+ck*vm1
      c00vm0=c00*vm0
      c11vm1=c11*vm1
      call vvmval(2,4,x,vm1,vm0)
      bvm2=bvm2+ck*vm0
      bvm3=bvm3+ck*vm1
 351  value=value+ck*(c00vm0+c11vm1+c20*vm0)
c      write(60,1847)bvm0,bvm1,bvm2,bvm3
      d22=(dx*dx-dy*dy)*k22
      d20=(3.0d0*dz*dz-1.0d0)*k20
      go to 36
c
c  xa---xxb
  171 bijk=bax*bax*bax
      c31=0.6d0*bax/ba
      c20=-0.6666666666666d0*bax
      go to 131
c  xa---xyb
  172 bijk=bax*bax*bay
      c31=0.2d0*bay/ba
      c20=-0.3333333333333d0*bay
      go to 131
c  xa---yyb
  173 bijk=bax*bay*bay
      c31=0.2d0*bax/ba
      c20=0.0d0
      go to 131
c  xa---xzb
  174 bijk=bax*bax*baz
      c31=0.2d0*baz/ba
      c20=-0.3333333333333d0*baz
      go to 131
c  xa---yzb
  175 bijk=bax*bay*baz
      c31=0.0d0
      c20=0.0d0
      go to 131
c  xa---zzb
  176 bijk=bax*baz*baz
      c31=0.2d0*bax/ba
      c20=0.0d0
      go to 131
c  ya---xxb
  181 bijk=bay*bax*bax
      c31=0.2d0*bay/ba
      c20=0.0d0
      go to 131
c  ya---xyb
  182 bijk=bay*bax*bay
      c31=0.2d0*bax/ba
      c20=-0.3333333333333d0*bax
      go to 131
c  ya---yyb
  183 bijk=bay*bay*bay
      c31=0.6d0*bay/ba
      c20=-0.6666666666666d0*bay
      go to 131
c  ya---xzb
  184 bijk=bay*bax*baz
      c31=0.0d0
      c20=0.0d0
      go to 131
c  ya---yzb
  185 bijk=bay*bay*baz
```

```
      c31=0.2d0*baz/ba
      c20=-0.3333333333333d0*baz
      go to 131
c  ya---zzb
  186 bijk=bay*baz*baz
      c31=0.2d0*bay/ba
      c20=0.0d0
      go to 131
c  za---xxb
  191 bijk=baz*bax*bax
      c31=0.2d0*baz/ba
      c20=0.0d0
      go to 131
c  za---xyb
  192 bijk=baz*bax*bay
      c31=0.0d0
      c20=0.0d0
      go to 131
c  za---yyb
  193 bijk=baz*bay*bay
      c31=0.2d0*baz/ba
      c20=0.0d0
      go to 131
c  za---xzb
  194 bijk=baz*bax*baz
      c31=0.2d0*bax/ba
      c20=-0.3333333333333d0*bax
      go to 131
c  za---yzb
  195 bijk=baz*bay*baz
      c31=0.2d0*bay/ba
      c20=-0.3333333333333d0*bay
      go to 131
c  za---zzb
  196 bijk=baz*baz*baz
      c31=0.6d0*baz/ba
      c20=-0.6666666666666d0*baz
      go to 131
c
c  xxa---xb
  201 bijk=bax*bax*bax
      c31=0.6d0*bax/ba
      c20=-0.3333333333333d0*bax
      bck=bax
      dk=dx
      delxk=1.0d0
      k22=0.5d0
      k20=-1.0d0/6.0d0
      go to 133
c  xxa---yb
  202 bijk=bax*bax*bay
      c31=0.2d0*bay/ba
      c20=-0.3333333333333d0*bay
      bck=bay
      dk=dy
      k22=0.5d0
      k20=-1.0d0/6.0d0
      go to 133
c  xxa---zb
  203 bijk=bax*bax*baz
      c31=0.2d0*baz/ba
      c20=-0.3333333333333d0*baz
      bck=baz
      dk=dz
```

```
      k22=0.5d0
      k20=-1.0d0/6.0d0
      go to 133
c  xya---xb
  204 bijk=bax*bay*bax
      c31=0.2d0*bay/ba
      c20=0.0d0
      go to 133
c  xya---yb
  205 bijk=bax*bay*bay
      c31=0.2d0*bax/ba
      c20=0.0d0
      go to 133
c  xya---zb
  206 bijk=bax*bay*baz
      c31=0.0d0
      c20=0.0d0
      go to 133
c  yya---xb
  207 bijk=bay*bay*bax
      c31=0.2d0*bax/ba
      c20=-0.3333333333333d0*bax
      bck=bax
      dk=dx
      k22=0.5d0
      k20=-1.0d0/6.0d0
      go to 133
c  yya---yb
  208 bijk=bay*bay*bay
      c31=0.6d0*bay/ba
      bck=bay
      dk=dy
      delyk=1.0d0
      k22=0.5d0
      k20=-1.0d0/6.0d0
      go to 133
c  yya---zb
  209 bijk=bay*bay*baz
      c31=0.2d0*baz/ba
      c20=-0.3333333333333d0*baz
      bck=baz
      dk=dz
      k22=0.5d0
      k20=-1.0d0/6.0d0
      go to 133
c  xza---xb
  210 bijk=bax*baz*bax
      c31=0.2d0*baz/ba
      c20=0.0d0
      go to 133
c  xza---yb
  211 bijk=bax*baz*bay
      c31=0.0d0
      c20=0.0d0
      go to 133
c  xza---zb
  212 bijk=bax*baz*baz
      c31=0.2d0*bax/ba
      c20=0.0d0
      go to 133
c  yza---xb
  213 bijk=bay*baz*bax
      c31=0.0d0
      c20=0.0d0
```

```
      go to 133
c  yza---yb
  214 bijk=bay*baz*bay
      c31=0.2d0*baz/ba
      c20=0.0d0
      go to 133
c  yza---zb
  215 bijk=bay*baz*baz
      c31=0.2d0*bay/ba
      c20=0.0d0
      go to 133
c  zza---xb
  216 bijk=baz*baz*bax
      c31=0.2d0*bax/ba
      c20=-0.3333333333333d0*bax
      bck=bax
      dk=dx
      k22=0.0d0
      k20=1.0d0/3.0d0
      go to 133
c  zza---yb
  217 bijk=baz*baz*bay
      c31=0.2d0*bay/ba
      c20=-0.3333333333333d0*bay
      bck=bay
      dk=dy
      k22=0.0d0
      k20=1.0d0/3.0d0
      go to 133
c  zza---zb
  218 bijk=baz*baz*baz
      c31=0.6d0*baz/ba
      c20=-0.3333333333333d0*baz
      bck=baz
      dk=dz
      delzk=1.0d0
      k22=0.0d0
      k20=1.0d0/3.0d0
      go to 133
c
  131 lstrt=ldsr2(kc)
      lstop=ldsp2(kc)
      lstrtl=lstr4(kc)
      lstopl=lstp4(kc)
      c00=0.0d0
      c11=bijk/ba
      c22t=c11/ba
      c22=-(2.0d0*c22t+c20)
      go to 134
  133 lstrtl=lstr4(kc)
      lstopl=lstp4(kc)
      lstrt=lstop
      c00=c20
      c11=0.0d0
      c22t=bijk/(ba*ba)
      c22=-(c22t+c20)
  134 c33=c22t/ba-c31
      c31=c31+c33
      c22=c22-5.0d0*c33*ail
      c20=c20+c22
      c11=c11-3.0d0*c22*ail
c
      do 1341k=lstrtl,lstopl
      zeta=xij+zlp(k)
```

```
      exa=exp(-d*(1.0d0-zeta1/xij))
      ck=-zval*clp(k)*exa
      x=zeta1*ba*dsqrt(zlp(k)/(zeta*xij))
      call vvmval(2,2,x,vm1,vm0)
      c11vm1=c11*vm1
      call vvmval(2,4,x,vm1,vm0)
1341  value=value+ck*(c20*vm0+c11vm1+c31*vm1)
c
      if( lstrt.eq.lstop)goto1321
      do 132k=lstrt,lstop
      zeta=xij+zlp(k)
      n=nlp(k)
      n1=n+1
      exa=exp(-d*(1.0d0-zeta1/zeta))
      srzi=1.0d0/dsqrt(zeta)
      ck=clp(k)*exa*(srzi**n)*0.5d0*ai1
      x=b*srzi
      call vvmval(1,n,x,vm,v)
      c31z=c31/zeta
      c20z=c20*srzi
      c11z=(c11+c20z*x+c31z*(x*x+n*0.5d0))*srzi
      c00z= (c20z+c31z*x)*n1*0.5d0*srzi
  132 value=value+(c00z*vm+c11z*v)*ck
      go to 290
 1321 if(lmax(kc).eq.0)goto290
      c11=-c00/ba
      lstrt=ldsr1(kc)
      lstop=ldsp1(kc)
      do 123k=lstrt,lstop
      zeta=xij+zlp(k)
      n=nlp(k)+2
      n1=n+1
      exa=exp(-d*(1.0d0-zeta1/zeta))
      srzi=1.0d0/dsqrt(zeta)
      ck=clp(k)*exa*(srzi**n)*0.5d0*ai1
      x=b*srzi
      call vvmval(1,n,x,vm,v)
  123 value=value+ck*(c00*vm+c11*srzi*v)
      if (lmax(kc).lt.3)goto290
      lstrt=ldsr3(kc)
      lstop=ldsp3(kc)
      do 125k=lstrt,lstop
      zeta=xij+zlp(k)
      n=nlp(k)
      n1=n+1
      exa=exp(-d*(1.0d0-zeta1/zeta))
      srzi=1.0d0/dsqrt(zeta)
      ck=clp(k)*exa*(srzi**n1)*0.5d0*ai1
      x=b*srzi
      d00=k22*(bck*(dy*dy-dx*dx)+ai1*(5.0d0*dk*(dy*dy-dx*dx)+
     #    2.0d0*(dx*delxk-dy*delyk)))
      dm11=-k22*(3.0d0/a)*(bck*(dy*dy-dx*dx)+ai1*(5.0d0*dk*
     #     (dy*dy-dx*dx)+2.0d0*(dx*delxk-dy*delyk)))
      d11=k22*dk*(dx*dx-dy*dy)
      d00=d00+k20*(bck*(1.0d0-3.0d0*dz*dz)-ai1*3.0d0*(5.0d0*dk
     #    *dz*dz-2.0d0*dz*delzk-dk))
      dm11=dm11-k20*3.0d0*ai1*(bck*(1.0d0-3.0d0*dz*dz)-ai1*3.0d0*
     #    (5.0d0*dk*dz*dz-2.0d0*dz*delzk-dk))
      d11=d11+k20*dk*(3.0d0*dz*dz-1.0d0)
      c00=n1*0.5d0*(d00+d11*srzi)/zeta
      c11= d00*a*0.5d0/zeta+dm11+
     #    d11*(a*a*0.25d0/zeta+n/2.0d0)/zeta
      call vvmval(1,n,x,vm,v)
  125 value=value+ck*(c00*vm+c11*v)
```

```
      go to 290
  505 go to (161,201,202,203),jtyp
      ai=bax
      aj=bax
      ia=1
      ja=1
      delij=1.0d0
      k22=0.50d0
      k20=-1.0d0/6.0d0
      delxi=1.0d0
      delxj=1.0d0
      go to 511
  506 go to (163,207,208,209),jtyp
      ai=bay
      aj=bay
      ia=2
      ja=2
      delij=1.0d0
      k22=0.50d0
      k20=-1.0d0/6.0d0
      delyi=1.0d0
      delyj=1.0d0
      go to 511
  507 go to (166,216,217,218),jtyp
      ai=baz
      aj=baz
      ia=3
      ja=3
      delij=1.0d0
      k22=0.0d0
      k20=1.0d0/3.0d0
      delzi=1.0d0
      delzj=1.0d0
      go to 511
  508 go to (162,204,205,206),jtyp
      ai=bax
      aj=bay
      ia=1
      ja=2
      delij=0.0d0
      go to 511
  509 go to (164,210,211,212),jtyp
      ai=bax
      aj=baz
      ia=1
      ja=3
      delij=0.0d0
      go to 511
  510 go to (165,213,214,215),jtyp
      ai=bay
      aj=baz
      ia=2
      ja=3
      delij=0.0d0
      go to 511
  511 jjtyp=jtyp-4
      dxk=0.0d0
      dxl=0.0d0
      dyk=0.0d0
      dyl=0.0d0
      dzk=0.0d0
      dzl=0.0d0
      go to (512,513,514,515,516,517),jjtyp
  512 ak=bax
```

```
      al=bax
      ka=1
      la=1
      dxk=1.0d0
      dxl=1.0d0
      delkl=1.0d0
      go to 231
 513  ak=bay
      al=bay
      ka=2
      la=2
      dyk=0.0d0
      dyl=0.0d0
      delkl=1.0d0
      go to 231
 514  ak=baz
      al=baz
      ka=3
      la=3
      dzk=1.0d0
      dzl=1.0d0
      delkl=1.0d0
      go to 231
 515  ak=bax
      al=bay
      ka=1
      la=2
      dxk=1.0d0
      dyl=1.0d0
      delkl=0.0d0
      go to 231
 516  ak=bax
      al=baz
      ka=1
      la=3
      dxk=1.0d0
      dzl=1.0d0
      delkl=0.0d0
      go to 231
 517  ak=bay
      al=baz
      ka=2
      la=3
      dyk=1.0d0
      dzl=1.0d0
      delkl=0.0d0
      go to 231
 231  delik=0.0d0
      delil=0.0d0
      deljk=0.0d0
      deljl=0.0d0
      if(ia.eq.ka)delik=1.0d0
      if(ia.eq.la)delil=1.0d0
      if(ja.eq.ka)deljk=1.0d0
      if(ja.eq.la)deljl=1.0d0
      aij=ai*aj
      akl=ak*al
      aijdkl=aij*delkl
      akldij=akl*delij
      aijkl=aij*akl
      bai=1.0d0/ba
      bai2=bai*bai
      aad=aijdkl+akldij+ai*(ak*deljl+al*deljk)+aj*(ak*delil+al*delik)
      dels=delij*delkl+delik*deljl+delil*deljk
```

```
      c20=akldij*0.3333333333333d0
      c22t=aijkl*bai2
      c22=c22t-c20
      c31=-(aad-aijdkl+akldij)*bai*0.2d0
      c33=-(2.0d0*c22t*bai+c31)
      c40=dels/15.0d0
      c42t=aad*bai2/7.0d0
      c42=c42t-dels/10.5d0
      c44=c22t*bai2-c42t+dels/35.0d0
      c42=c42+c44
      c33=c33-7.0d0*c44*ail
      c40=c40+c42
      c31=c31+c33-3.0d0*c42*ail
      c22=c22-5.0d0*c33*ail
      c20=c20+c22
      c11=  -3.0d0*c22*ail
      lstrt=lstr4(kc)
      lstop=lstp4(kc)
      bvm0=0.0d0
      bvm1=0.0d0
      do 2321k=lstrt,lstop
      zeta=xij+zlp(k)
      exa=exp(-d*(1.0d0-zetal/xij))
      ck=-zval*clp(k)*exa
      x=zetal*ba*dsqrt(zlp(k)/(zeta*xij))
      call vvmval(2,2,x,vm1,vm0)
      c11vm1=c11*vm1
      call vvmval(2,4,x,vm1,vm0)
      c20vm0=c20*vm0
      c31vm1=c31*vm1
      call vvmval(2,6,x,vm1,vm0)
 2321 value=value+ck*(c20vm0+c11vm1+c31*vm1+c40*vm0)
c
c      do 232 k=lstrt,lstop
c      zeta=xij+zlp(k)
c      n=nlp(k)
c      n1=n+1
c      exa=exp(-d*(1.0d0-zetal/zeta))
c      srzi=1.0d0/dsqrt(zeta)
c      ck=clp(k)*exa*(srzi**n )*0.5d0*ail
c      x=b*srzi
c      call vvmval(1,n,x,vm,v)
c      c40z=c40*srzi
c      c31z=(c31+c40z*x)/zeta
c      c20z=(c20+c40z*srzi*(n1+2)*0.5d0)*srzi
c      c11z=(c11+c20z*x+c31z*(x*x+n*0.5d0))*srzi
c      c00z=    (c20z+c31z*x)*n1*0.5d0*srzi
c  232 value=value+(c00z*vm   +c11z*v    )*ck
      if(delij.eq.0.0d0)goto289
      if(lmax(kc).eq.0)goto290
      c00=akl*0.3333333333333d0
      c11=-c00*bai*2.0d0
      c20=delkl*0.1111111111111d0
      c22=c00*bai2-c20
      c20=c20+c22
      c11=c11-3.0d0*c22*ail
c
c     a<ri*ri/vs/rk*rl>b => r*r*<s/vs/ri*rj>b
c
      lstrt=ldsr1(kc)
      lstop=ldsp1(kc)
      do 124k=lstrt,lstop
      zeta=xij+zlp(k)
      n=nlp(k)+2
```

```
      n1=n+1
      exa=exp(-d*(1.0d0-zeta1/zeta))
      srzi=1.0d0/dsqrt(zeta)
      ck=clp(k)*exa*(srzi**n)*0.5d0*ai1
      x=b*srzi
      call vvmval(1,n,x,vm,v)
  124 value=value+((c00+c20*n1*0.5d0/zeta)*vm+(c11+c20*x*srzi)*
     #  srzi*v    )*ck
      if(lmax(kc).lt.3)goto290
c
c   a<ri*ri/vd/rk*rl>b => r*r*(k22*<y22/vd/rk*rl>b+k20*<y20/vd/rk*rl>b
c
      d00=k22*(ak*al*ba**2*a**2*bax**2-ak*al*ba**2*a**2*bay**2+
     # 10.0d0*ak*al*ba*a*bax**2-10.0d0*ak*al*ba*a*bay**2+35.0d0*ak*
     # al*bax**2 - 35.d0*ak*al*bay**2 -
     # 2.d0*ak*ba**3*a*bax*dxl + 2.d0*ak*ba**3*a*bay*dyl -
     # 10.0d0*ak*ba**2*bax*dxl + 10.0d0*ak*ba**2*bay*dyl -
     # 2.d0*al*ba**3*a*bax*dxk + 2.d0*al*ba**3*a*bay*dyk -
     # 10.0d0*al*ba**2*bax*dxk + 10.0d0*al*ba**2*bay*dyk +
     # 2.d0*ba**4*dxk*dxl - 2.d0*ba**4*dyk*dyl-2.d0*ba**3*dkl*a*bax**2+
     # 2.d0*ba**3*dkl*a*bay**2 - 5.d0*ba**2*dkl*bax**2 +
     # 5.d0*ba**2*dkl*bay**2 ) / ( ba**4*a**2 )
c
c
      dm11=-k22*(3.d0*ak*al*ba**2*a**2*bax**2-
     # 3.d0*ak*al*ba**2*a**2*bay**2 + 30.0d0*ak*al*ba*a*bax**2 -
     # 30.0d0*ak*al*ba*a*bay**2 + 105.d0*ak*al*bax**2 -
     # 105.d0*ak*al*bay**2 - 6.d0*ak*ba**3*a*bax*dxl +
     # 6.d0*ak*ba**3*a*bay*dyl - 30.0d0*ak*ba**2*bax*dxl +
     # 30.0d0*ak*ba**2*bay*dyl - 6.d0*al*ba**3*a*bax*dxk +
     # 6.d0*al*ba**3*a*bay*dyk - 30.0d0*al*ba**2*bax*dxk +
     # 30.0d0*al*ba**2*bay*dyk - ba**4*dkl*a**2*bax**2 +
     # ba**4*dkl*a**2*bay**2 + 6.d0*ba**4*dxk*dxl -
     # 6.d0*ba**4*dyk*dyl - 6.d0*ba**3*dkl*a*bax**2 +
     # 6.d0*ba**3*dkl*a*bay**2 - 15.d0*ba**2*dkl*bax**2 +
     # 15.d0*ba**2*dkl*bay**2 ) / ( ba**4*a**3 )
c
c
      d11=-k22*(2.d0*ak*al*ba*a*bax**2-2.d0*ak*al*ba*a*bay**2+
     # 10.0d0*ak*al*bax**2-10.0d0*ak*al*bay**2-2.d0*ak*ba**2*bax*dxl+
     # 2.d0*ak*ba**2*bay*dyl - 2.d0*al*ba**2*bax*dxk +
     # 2.d0*al*ba**2*bay*dyk - ba**2*dkl*bax**2 + ba**2*dkl*bay**2 ) /
     # ( ba**4*a )
c
c
      d20=k22*ak*al*(bax**2-bay**2)/(ba**4)
c
c
c  d20
c
      d00=d00-k20*(ak*al*ba**2*a**2*bax**2+ak*al*ba**2*a**2*bay**2-
     # 2.d0*ak*al*ba**2*a**2*baz**2 + 10.0d0*ak*al*ba*a*bax**2 +
     # 10.0d0*ak*al*ba*a*bay**2 - 20.0d0*ak*al*ba*a*baz**2 +
     # 35.d0*ak*al*bax**2 + 35.d0*ak*al*bay**2 - 70.0d0*ak*al*baz**2-
     # 2.d0*ak*ba**3*a*bax*dxl - 2.d0*ak*ba**3*a*bay*dyl +
     # 4.d0*ak*ba**3*a*baz*dzl - 10.0d0*ak*ba**2*bax*dxl -
     # 10.0d0*ak*ba**2*bay*dyl + 20.0d0*ak*ba**2*baz*dzl -
     # 2.d0*al*ba**3*a*bax*dxk - 2.d0*al*ba**3*a*bay*dyk +
     # 4.d0*al*ba**3*a*baz*dzk - 10.0d0*al*ba**2*bax*dxk -
     # 10.0d0*al*ba**2*bay*dyk + 20.0d0*al*ba**2*baz*dzk +
     # 2.d0*ba**4*dxk*dxl + 2.d0*ba**4*dyk*dyl - 4.d0*ba**4*dzk*dzl -
     # 2.d0*ba**3*dkl*a*bax**2 - 2.d0*ba**3*dkl*a*bay**2 +
     # 4.d0*ba**3*dkl*a*baz**2 - 5.d0*ba**2*dkl*bax**2 -
     # 5.d0*ba**2*dkl*bay**2 + 10.0d0*ba**2*dkl*baz**2 )/(ba**4*a**2)
```

```
c
c
      dml1=dml1+k20*(3.d0*ak*al*ba**2*a**2*bax**2+
     # 3.d0*ak*al*ba**2*a**2*bay**2 - 6.d0*ak*al*ba**2*a**2*baz**2 +
     # 30.0d0*ak*al*ba*a*bax**2 + 30.0d0*ak*al*ba*a*bay**2 -
     # 60.0d0*ak*al*ba*a*baz**2 + 105.d0*ak*al*bax**2 +
     # 105.d0*ak*al*bay**2 - 210.0d0*ak*al*baz**2 -
     # 6.d0*ak*ba**3*a*bax*dxl - 6.d0*ak*ba**3*a*bay*dyl +
     # 12.d0*ak*ba**3*a*baz*dzl - 30.0d0*ak*ba**2*bax*dxl -
     # 30.0d0*ak*ba**2*bay*dyl + 60.0d0*ak*ba**2*baz*dzl -
     # 6.d0*al*ba**3*a*bax*dxk - 6.d0*al*ba**3*a*bay*dyk +
     # 12.d0*al*ba**3*a*baz*dzk - 30.0d0*al*ba**2*bax*dxk -
     # 30.0d0*al*ba**2*bay*dyk + 60.0d0*al*ba**2*baz*dzk -
     # ba**4*dkl*a**2*bax**2 - ba**4*dkl*a**2*bay**2 +
     # 2.d0*ba**4*dkl*a**2*baz**2 + 6.d0*ba**4*dxk*dxl +
     # 6.d0*ba**4*dyk*dyl - 12.d0*ba**4*dzk*dzl -
     # 6.d0*ba**3*dkl*a*bax**2 - 6.d0*ba**3*dkl*a*bay**2 +
     # 12.d0*ba**3*dkl*a*baz**2 - 15.d0*ba**2*dkl*bax**2 -
     # 15.d0*ba**2*dkl*bay**2 + 30.0d0*ba**2*dkl*baz**2 ) /
     # ( ba**4*a**3 )
c
c
c
      dl1=dl1+k20*(2.d0*ak*al*ba*a*bax**2+2.d0*ak*al*ba*a*bay**2-
     # 4.d0*ak*al*ba*a*baz**2 + 10.0d0*ak*al*bax**2 +
     # 10.0d0*ak*al*bay**2 - 20.0d0*ak*al*baz**2 -
     # 2.d0*ak*ba**2*bax*dxl - 2.d0*ak*ba**2*bay*dyl +
     # 4.d0*ak*ba**2*baz*dzl - 2.d0*al*ba**2*bax*dxk -
     # 2.d0*al*ba**2*bay*dyk + 4.d0*al*ba**2*baz*dzk -
     # ba**2*dkl*bax**2 - ba**2*dkl*bay**2 + 2.d0*ba**2*dkl*baz**2 )
     # / ( ba**4*a )
c
c
c
      d20=d20-k20*ak*al*(bax**2+bay**2-2.d0*baz**2)/(ba**4)
c
      lstrt=ldsr3(kc)
      lstop=ldsp3(kc)
      do 237k=lstrt,lstop
      zeta=xij+zlp(k)
      n=nlp(k)+2
      n1=n+1
      exa=exp(-d*(1.0d0-zetal/zeta))
      srzi=1.0d0/dsqrt(zeta)
      ck=clp(k)*exa*(srzi**n)*0.5d0*ail
      x=b*srzi
      call vvmval(1,n,x,vm,v)
  237 value=value+((d00+dl1*srzi+d20*(b*srzi+(n+3)/2.0d0)/(2.0d0
     #    *zeta))*(n+1)*srzi*vm/2.0d0 +(d00*b/(2.0d0*zeta)+dml1+
     #    dl1*(b*b/(2.0d0*zeta)+n)+d20*b*(b*b/(4.0d0*zeta)+n+1.5d0)
     #    /(2.0d0*zeta*zeta))*v )*ck
      go to 290
c
c     a<ri*rj/vd/rk*rl>b , i .ne. j => r*r*<yij/vd/rk*rl>b
c     where yij=x*y/r*r , x*z/r*r or y*z/r*r.
c
 289  d00=( ak*al*ba**2*a**2*ai*aj+10.0d0*ak*al*ba*a*ai*aj+
     # 35.d0*ak*al*ai*aj - ak*ba**3*a*ai*deljl - ak*ba**3*a*aj*delil -
     # 5.d0*ak*ba**2*ai*deljl - 5.d0*ak*ba**2*aj*delil -
     # al*ba**3*a*ai*deljk - al*ba**3*a*aj*delik -
     # 5.d0*al*ba**2*ai*deljk - 5.d0*al*ba**2*aj*delik +
     # ba**4*delik*deljl + ba**4*delil*deljk -
     # 2.d0*ba**3*delkl*a*ai*aj - 5.d0*ba**2*delkl*ai*aj )/(ba**4*a**2)
c
```

```
c# dimll
c
      dmll=-(3.d0*ak*al*ba**2*a**2*ai*aj+
     # 30.0d0*ak*al*ba*a*ai*aj + 105.d0*ak*al*ai*aj -
     # 3.d0*ak*ba**3*a*ai*deljl - 3.d0*ak*ba**3*a*aj*delil -
     # 15.d0*ak*ba**2*ai*deljl - 15.d0*ak*ba**2*aj*delil -
     # 3.d0*al*ba**3*a*ai*deljk - 3.d0*al*ba**3*a*aj*delik -
     # 15.d0*al*ba**2*ai*deljk - 15.d0*al*ba**2*aj*delik -
     # ba**4*delkl*a**2*ai*aj + 3.d0*ba**4*delik*deljl +
     # 3.d0*ba**4*delil*deljk - 6.d0*ba**3*delkl*a*ai*aj -
     # 15.d0*ba**2*delkl*ai*aj ) / ( ba**4*a**3 )
c
c# dill
c
      dll=-(2.d0*ak*al*ba*a*ai*aj+10.0d0*ak*al*ai*aj-
     # ak*ba**2*ai*deljl - ak*ba**2*aj*delil - al*ba**2*ai*deljk -
     # al*ba**2*aj*delik - ba**2*delkl*ai*aj ) / ( ba**4*a )
c
c# di20
c
      d20=ak*al*ai*aj/(ba**4)
c
c@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
c
      lstrt=ldsr3(kc)
      lstop=ldsp3(kc)
      do 288k=lstrt,lstop
      zeta=xij+zlp(k)
      n=nlp(k)
      nl=n+1
      exa=exp(-d*(1.0d0-zetal/zeta))
      srzi=1.0d0/dsqrt(zeta)
      ck=clp(k)*exa*(srzi**n)*0.5d0*ail
      x=b*srzi
      call vvmval(1,n,x,vm,v)
  288 value=value+((d00+dll*srzi+d20*(a*srzi+n+3.0d0)/(2.0d0
     #    *zeta) )*nl*srzi*vm/2.0d0 +(d00*a/(2.0d0*zeta) + dmll +
     #    dll * (a*a/(2.0d0*zeta)+n) /(2.0d0*zeta) +
     #    d20 * a*(a*a/(4.0d0*zeta)+n+1.5d0)
     #    /(2.0d0*zeta*zeta))*v )*ck
c
  290 xbaa=value
  291 return
    1 write(60,2)
    2 format(1hl,///,20x,18hfatal purity error)
      stop
      end
      subroutine vbca
c    version #1 feb, 1985
c
      implicit double precision(a-h,o-z)
      dimension si00(5),simll(5),pi00(5),pimll(5),pill(5),di00(5),
     #          dimll(5),dill(5),di20(5),sj00(5),sjmll(5),pj00(5),
     #          pjmll(5),pjll(5),dj00(5),djmll(5),djll(5),dj20(5)
c
c  common block for ddmx test
c
      common/test/ddtest
      common/bfcom/xcab,ityp,jtyp,zetl,zet2,kc,idum,caa(3),ca,
     #  baa(3)     ,ba,daa(3)     ,dasq,phase,zval
      common/int2/nlp(200),clp(200),zlp(200),lstrl(20),lstr2(20),
     1 lstr3(20),lstr4(20),lstpl(20),lstp2(20),lstp3(20),lstp4(20),
     1 ldsrl(20),ldsr2(20),ldsr3(20),ldspl(20),ldsp2(20),ldsp3(20),
     1 dsmx(20),dpmx(20),ddmx(20),xmx(20),lmax(20)
```

```
      common/cnl112/cm211,cm110,cm101,c000,c011,c101,c110,c200,c211,
     # c301,c310,c400,c411,xij,aht,bht,d,inttyp,ldstr,ldstp
      common/aaacom/zeta
      common/erfp/b,xijp
c     data fourpi/12.56637060d0/   deleted
c     data srpi4/0.4431134627265d0/  deleted
      data srpi/1.77245385090d0/
      data pi/3.14159265358979323846d0/
c
c
c  initialization of parameters
c
      ia=0
      ja=0
      ka=0
      la=0
c
      xij=zet1+zet2
      xijp=xij
      lm=lmax(kc)+1
      lstrlm=lstr4(kc)
      lstplm=lstp4(kc)
    3 delij=0.0d0
      value=0.0d0
      aht=ca*zet1
      bht=ba*zet2
      x1=aht*2.0d0
      x2=bht*2.0d0
      d=-0.5d0*(x1*ca+x2*ba)
c
      go to (11,12,13,14,15,16,17,18,19,20),jtyp
      go to 4
   11 inttyp=1
      go to 100
   12 ia=1
      go to 21
   13 ia=2
      go to 21
   14 ia=3
      go to 21
   15 ia=1
      ja=1
      delij=0.333333333333333d0
      go to 27
   16 ia=2
      ja=2
      delij=0.333333333333333d0
      go to 27
   17 ia=3
      ja=3
      delij=0.333333333333333d0
      go to 27
   18 ia=1
      ja=2
      delij=0.0d0
      go to 27
   19 ia=1
      ja=3
      delij=0.0d0
      go to 27
   20 ia=2
      ja=3
      delij=0.0d0
      go to 27
```

```
21 go to (22,23,24,25),ityp
22 ai=baa(ia)
   inttyp=2
   go to 100
23 ja=1
   go to 26
24 ja=2
   go to 26
25 ja=3
   go to 26
26 ai=baa(ia)
   aj=caa(ja)
   inttyp=3
   aij=ai*aj
   delij=0.0d0
   if(ia.eq.ja)delij=0.333333333333333d0
   go to 100
27 go to (31,32,33,34,35,36,37,38,39,40),ityp
28 ai=baa(ia)
   aj=baa(ja)
   ak=caa(ka)
   aij=ai*aj
   aijk=aij*ak
   inttyp=5
   delik=0.0d0
   deljk=0.0d0
   if(ia.eq.ka)delik=0.333333333333333d0
   if(ja.eq.ka)deljk=0.333333333333333d0
   c2t=(ai*deljk+aj*delik+ak*delij)
   go to 100
29 ai=baa(ia)
   aj=baa(ja)
   ak=caa(ka)
   al=caa(la)
   inttyp=6
   aij=ai*aj
   aik=ai*ak
   ail=ai*al
   ajk=aj*ak
   ajl=aj*al
   akl=ak*al
   aijkl=aij*akl
   delik=0.0d0
   delil=0.0d0
   deljk=0.0d0
   deljl=0.0d0
   if(ia.eq.ka)delik=0.333333333333333d0
   if(ia.eq.la)delil=0.333333333333333d0
   if(ja.eq.ka)deljk=0.333333333333333d0
   if(ja.eq.la)deljl=0.333333333333333d0
   dels=delij*delkl+delik*deljl+delil*deljk
   c2t=  aik*deljl+ail*deljk+ajk*delil+ajl*delik
   go to 100
31 ai=baa(ia)
   aj=baa(ja)
   aij=ai*aj
   inttyp=4
   go to 100
32 ka=1
   go to 28
33 ka=2
   go to 28
34 ka=3
   go to 28
```

```
   35 ka=1
      la=1
      delkl=0.333333333333333d0
      go to 29
   36 ka=2
      la=2
      delkl=0.333333333333333d0
      go to 29
   37 ka=3
      la=3
      delkl=0.333333333333333d0
      go to 29
   38 ka=1
      la=2
      delkl=0.0d0
      go to 29
   39 ka=1
      la=3
      delkl=0.0d0
      go to 29
   40 ka=2
      la=3
      delkl=0.0d0
      go to 29
c
  100 if (dasq.gt.1.0d-16)goto200
c
      go to (101,102,103,103,105,106),inttyp
  101 c0=1.0d0
      go to 160
  102 c0=-ai
      go to 160
  103 c0=aij
      c2=delij
      go to 170
  105 c0=-aijk
      c2=-c2t
      go to 170
c
c
  160 if(c0.eq.0)goto400
      vc0=0.0d0
      do 163k=lstrlm,lstplm
      zeta=xij+zlp(k)
      ck=clp(k)
      n=nlp(k)
      vc0=vc0-zval*clp(k)*dsqrt(zlp(k)/zeta)/(2.0d0*xij)
c       vc0=vc0+fa(n)*ck
  163 continue
      xcab=phase*c0*vc0
      go to 400
c
  170 if(c0.eq.0.0d0.and.c2.eq.0.0d0)goto400
      vc0=0.0d0
      vc2=0.0d0
      do 173k=lstrlm,lstplm
      zeta=xij+zlp(k)
      ck=clp(k)
      n=nlp(k)
      if(c2.eq.0.0d0)goto172
      vc2=vc2-zval*ck*dsqrt(zlp(k)/zeta)*(3.0d0*xij
     #    +2.0d0*clp(k))/(4.0d0*xij*xij*zeta)
c       vc2=vc2+fa(n+2)*ck
  172 if(c0.eq.0.0d0)goto173
```

```
      vc0=vc0-zval*clp(k)*dsqrt(zlp(k)/zeta)/(2.0d0*xij)
c      vc0=vc0+fa(n)*ck
  173 continue
      xcab=phase*(c2*vc2+c0*vc0)
      go to 400
c
c
  106 c0=aijkl
      c2  =(aij*delkl+c2t+akl*delij)
      c4=dels*0.6d0
      vc0=0.0d0
      vc2=0.0d0
      vc4=0.0d0
      do 183k=lstrlm,lstplm
      zeta=xij+zlp(k)
      ck=clp(k)
      n=nlp(k)
      vc4=vc4-zval*ck*dsqrt(zlp(k)/pi)*(fa(4)+2.0d0*
     1 (fa(2)+fa(0)/xij)/xij)/xij
c      vc4=vc4+fa(n+4)*ck
      if(c2.eq.0.0d0)goto182
      vc2=vc2-zval*ck*dsqrt(zlp(k)/zeta)*(3.0d0*xij
     #     +2.0d0*clp(k))/(4.0d0*xij*xij*zeta)
c      vc2=vc2+fa(n+2)*ck
  182 if(c0.eq.0.0d0)goto183
      vc0=vc0-zval*clp(k)*dsqrt(zlp(k)/zeta)/(2.0d0*xij)
c      vc0=vc0+fa(n)*ck
  183 continue
      xcab=phase*(c2*vc2+c0*vc0+c4*vc4)
      go to 400
c
c
  200 da=dsqrt(dasq)
      b=da*xij
      a=b*2.0d0
      ail=1.0d0/a
      dl=b*da
      srab=dsqrt(xij)
      go to (210,220,230,230,240,250),inttyp
c  sc---sb
  210 c00=1.0d0
  211 vc00=0.0d0
      do 212k=lstrlm,lstplm
      zeta=xij+zlp(k)
      n=nlp(k)
      exa=dsqrt(zlp(k)/(xij+zlp(k)))
      x=da*srab*exa
      call vvmval(2,n,x,vml,vm0)
      vc00=vc00-zval*clp(k)*vm0
  212 continue
c      write(60,218)vc00
c  218 format(1x,' vbca <s/core/s>= ',d15.8)
c      if((lstrlm+2) .eq. lstplm) goto 219
c      do 213 k=lstrlm+2,lstplm
c      zeta=xij+zlp(k)
c      n=nlp(k)
c      exa=exp(-dl*(1.0d0-xij/zeta))
c      srzi=1.0d0/dsqrt(zeta)
c      ck=clp(k)*exa*(srzi**n )*0.5d0*ail
c      x=b*srzi
c      call vvmval(0,n,x,vm,v)
c  213 vc00 =vc00+ck*vm
      xcab=phase*vc00
c      write(60,2227)xcab
```

```
c2227  format(1x,' exiting <s|vmax|s> xcab=',f14.8)
       go to 400
c  sc---pb
  220 c00=-ai
      c11=daa(ia)/da
  221 if(c00.eq.0.0d0.and.c11.eq.0.0d0)goto400
      vc00=0.0d0
      vc11=0.0d0
      do 222k=lstrlm,lstplm
      zeta=xij+zlp(k)
      exa=dsqrt(zlp(k)/zeta)
      x=da*srab*exa
      call vvmval(2,2,x,vm1,vm0)
      vc00=vc00-zval*clp(k)*vm0
      if(c11)223,222,223
  223 vc11=vc11-zval*clp(k)*vm1
  222 continue
c      write(60,11992)vc00,vc11
c11992 format(1x,' m0 = ',d15.8,' r*m1 = ',d15.8)
      xcab=phase*(c00*vc00+c11*vc11)
c      write(60,1829)xcab
c 1829 format(1x,' vbca <s/core/p>=',d15.8)
      go to 400
c  pc---pb or sc---db
  230 di=daa(ia)/da
      dj=daa(ja)/da
      c20=delij
      value=0.0d0
      c00=aij
      c11=-(di*aj+dj*ai)
      c22=di*dj-c20
      c20=c20+c22
      c11=c11-3.0d0*c22*ai1
      if(c00.eq.0.0d0.and.c20.eq.0.0d0.and.c11.eq.0.0d0)goto400
      bvm1=0.0d0
      bvm0=0.0d0
      bvm2=0.0d0
      bvm3=0.0d0
      do 232k=lstrlm,lstplm
      zeta=xij+zlp(k)
      exa=dsqrt(zlp(k)/zeta)
      x=da*srab*exa
      call vvmval(2,2,x,vm1,vm0)
      c00vm0=c00*vm0
      c11vm1=c11*vm1
      bvm0=bvm0-zval*clp(k)*vm0
      bvm1=bvm1-zval*clp(k)*vm1
c      write(60,2714)clp(k),zlp(k),x
c 2714 format(1x,' clp(k)=',d15.8,' zlp(k)=',d15.8,' x=',d15.8)
      call vvmval(2,4,x,vm1,vm0)
      bvm2=bvm2-zval*clp(k)*vm0
      bvm3=bvm3-zval*clp(k)*vm1
232   value=value-zval*clp(k)*(c00vm0+c20*vm0+c11vm1)
c      write(60,18823)bvm0,bvm1,bvm2,bvm3
c18823 format(1x,' m0 = ',d15.8,' r*m1 = ',d15.8
c     # ,/,' r*r*m0 = ',d15.8,' r*r*r*m1= ',d15.8)
      xcab=phase*value
c      write(60,7251)xcab
c7251  format(1x,' <p/core/p> =',d15.8)
      go to 400
c  pc---db
  240 di=daa(ia)/da
      dj=daa(ja)/da
      dk=daa(ka)/da
```

```
      c00=-aijk
      c20=-c2t
      c31=(dk*delij+dj*delik+di*deljk)*0.6d0
      c33=di*dj*dk-c31
      c22=-(aj*di*dk+ai*dj*dk+ak*di*dj+c20)
      c11=ai*aj*dk+ak*aj*di+ak*ai*dj
      c31=c31+c33
      c22=c22-5.0d0*c33*ail
      c20=c20+c22
      c11=c11-3.0d0*c22*ail
      value=0.0d0
      do 243k=lstrlm,lstplm
      zeta=xij+zlp(k)
      exa=dsqrt(zlp(k)/zeta)
      x=da*srab*exa
      call vvmval(2,2,x,vm1,vm0)
      c00vm0=c00*vm0
      c11vm1=c11*vm1
      call vvmval(2,4,x,vm1,vm0)
      value=value-zval*clp(k)*(c00vm0+c11vm1+c20*vm0+c31*vm1)
 243  continue
c      do 242 k=lstrlm,lstplm
c      n=nlp(k)
c      zeta=xij+zlp(k)
c      n1=n+1
c      exa=exp(-d1*(1.0d0-xij/zeta))
c      srzi=1.0d0/dsqrt(zeta)
c      ck=clp(k)*exa*(srzi**n )*0.5d0*aibca3070
c      srzi=1.0d0/dsqrt(zeta)
c      ck=clp(k)*exa*(srzi**n )*0.5d0*ail
c      x=b*srzi
c      call vvmval(1,n,x,vm,v)
c      c31z=c31/zeta
c      c20z=c20*srzi
c      c11z=(c11+c20z*x+c31z*(x*x+n*0.5d0))*srzi
c      c00z=c00+(c20z+c31z*x)*n1*0.5d0*srzi
c  242 value=value+(c00z*vm   +c11z*v    )*ck
      xcab=phase*value
      go to 400
c  dc---db
  250 dai=1.0d0/da
      di=daa(ia)*dai
      dj=daa(ja)*dai
      dk=daa(ka)*dai
      dl=daa(la)*dai
      dij=di*dj
      dik=di*dk
      dil=di*dl
      djk=dj*dk
      djl=dj*dl
      dkl=dk*dl
      adij=ai*dj
      adkl=ak*dl
      daij=di*aj
      dakl=dk*al
      addaij=adij+daij
      addakl=adkl+dakl
      c00=aijkl
      c11=-(aij*addakl+akl*addaij)
      c20 =(aij*delkl+c2t+akl*delij)
      c22t=aij*dkl+aik*djl+ail*djk+ajk*dil+ajl*dik+akl*dij
      c22=c22t-c20
      c31=-(delij*addakl+delkl*addaij+delik*(aj*dl+al*dj)+delil*(aj*dk+
     #  ak*dj)+deljk*(ai*dl+al*di)+deljl*(ai*dk+ak*di))*0.6d0
```

```
      c33=-(dij*addakl+dkl*addaij+c31)
      c40=dels*0.6d0
      c42t=(delij*dkl+delik*djl+delil*djk+deljk*dil+deljl*dik+delkl*dij)
      c42=(c42t-dels*2.0d0)*0.4285714285714d0
      c44=dij*dkl+(dels*0.6d0-c42t)*0.4285714285714d0
      c42=c42+c44
      c33=c33-7.0d0*c44*ail
      c40=c40+c42
      c31=c31+c33-3.0d0*c42*ail
      c22=c22-5.0d0*c33*ail
      c20=c20+c22
      c11=c11-3.0d0*c22*ail
      value=0.0d0
      do 353k=lstrlm,lstplm
      zeta=xij+zlp(k)
      exa=dsqrt(zlp(k)/zeta)
      x=da*srab*exa
      call vvmval(2,2,x,vm1,vm0)
      c00vm0=c00*vm0
      c11vm1=c11*vm1
      call vvmval(2,4,x,vm1,vm0)
      c20vm0=c20*vm0
      c31vm1=c31*vm1
      call vvmval(2,6,x,vm1,vm0)
      value=value-zval*clp(k)*(c00vm0+c11vm1+c20vm0+c31vm1+c40*vm0)
 353  continue
c      do 352 k=lstrlm,lstplm
c      n=nlp(k)
c      zeta=xij+zlp(k)
c      n1=n+1
c      exa=exp(-d1*(1.0d0-xij/zeta))
c      srzi=1.0d0/dsqrt(zeta)
c      ck=clp(k)*exa*(srzi**n )*0.5d0*ail
c      x=b*srzi
c      call vvmval(1,n,x,vm,v)
c      c40z=c40*srzi
c      c31z=(c31+c40z*x)/zeta
c      c20z=(c20+c40z*srzi*(n1+2)*0.5d0)*srzi
c      c11z=(c11+c20z*x+c31z*(x*x+n*0.5d0))*srzi
c      c00z=c00+(c20z+c31z*x)*n1*0.5d0*srzi
c  352 value=value+(c00z*vm   +c11z*v    )*ck
      xcab=phase*value
      go to 400
c
  400 if(lmax(kc).lt.1)return
      if(dsmx(kc)+d)700,700,401
  401 continue
      ldstr =ldsrl(kc)
      ldstp =ldspl(kc)
      go to (410,402,430,404,405,406),inttyp
  402 c00b=-ai
      go to 420
  404 c00b=aij
      c20b=c00b/(ba*ba)
      c11b=-2.0d0*c00b/ba-3.0d0*(c20b-delij)/x2
      if(c00b.eq.0.0d0.and.c11b.eq.0.0d0)goto700
      go to 435
  405 c00b=aij
      c20b=c00b/(ba*ba)
      c11b=-2.0d0*c00b/ba-3.0d0*(c20b-delij)/x2
      if(c00b.eq.0.0d0.and.c11b.eq.0.0d0)goto700
      go to 440
  406 c00b=aij
      c20b=c00b/(ba*ba)
```

```
      c11b=-2.0d0*c00b/ba-3.0d0*(c20b-delij)/x2
      if(c00b.eq.0.0d0.and.c11b.eq.0.0d0)goto700
      go to 450
c  sc--s--sb
  410 c000=1.0d0
      go to 500
c  sc--s--pb
  420  c000=c00b
      if(c00b)423,700,423
  423 c101=-c00b/ba
      go to 500
c  pc--s--pb
  430 c000=aij
  431 if(c000)433,700,433
  433 c101=-c000/ba
      c110=-c000/ca
      c211=-c101/ca
      go to 500
c  sc--s--db
  435 c000=c00b
      c200=c20b
      c101=c11b
      go to 500
c  pc--s--db
  440 c00c=-ak
      if(c00c.eq.0.0d0)goto700
      c000=c00c*c00b
      c101=c00c*c11b
      c200=c00c*c20b
      c110=-c000/ca
      c211=-c101/ca
      c310=-c200/ca
      go to 500
c  dc--s--db
  450 c00c=akl
      c20c=c00c/(ca*ca)
      c11c=-2.0d0*c00c/ca-3.0d0*(c20c-delkl)/x1
      if(c00c.eq.0.0d0.and.c11c.eq.0.0d0)goto700
      c000=c00c*c00b
      c200=c00c*c20b+c20c*c00b
      c301=c20c*c11b
      c310=c11c*c20b
      c101=c00c*c11b
      c110=c11c*c00b
      c400=c20c*c20b
      c211=c11c*c11b
      go to 500
c
  500 call vbcas(xcab)
c      write(60,3427)xcab
c3427  format(1x,' xcab after <s|s>',d15.9)
c
  700 if(lmax(kc).lt.2)return
      if(dpmx(kc)+d*(ca+ba)**2)1925,1925,701
  701 ldstr =ldsr2(kc)
      ldstp =ldsp2(kc)
      cba=ca*ba
      cbd=(caa(1)*baa(1)+caa(2)*baa(2)+caa(3)*baa(3))/cba
      go to (710,720,730,740,750,760),inttyp
c
c  sc--p--sb
  710 c011=cbd
  711 if(c011)713,1925,713
  713 vc011=0.0d0
```

```
      go to 800
c  sc--p--pb
  720 c011=-ai*cbd
      c110=-c011/ba
      c011=c011-3.0d0*(c110-caa(ia)*0.333333333333333d0/ca)/x2
      if(c011.eq.0.0d0.and.c110.eq.0.0d0)goto1925
      go to 800
c  pc--p--pb
  730 c200=delij*0.333333333333333d0
      baa2=baa(ja)*ai
      caa2=caa(ia)*aj
      c101=-baa2/(3.0d0*ba*ba)
      c110=-caa2/(3.0d0*ca*ca)
      cc200=aij*cbd/cba
      if(c200)729,728,729
  728 if(c101.eq.0.0d0.and.c110.eq.0.0d0.and.cc200.eq.0.0d0)goto1925
  729 continue
      d110=3.0d0*(cc200+c101)/x1
      d101=3.0d0*(cc200+c110)/x2
       c011=cba*cc200+d110*ba+d101*ca+9.0d0*(cc200+c200+c110+c101)/(x1*
     #  x2)
       c110=-ca*cc200-d110
       c101=-ba*cc200-d101
      c200=cc200
      go to 800
c  sc--p--db
  740 c011=aij*cbd
      c110=-(caa(ia)*aj+caa(ja)*ai)/(3.0d0*ca)
      c211=-0.6d0*(c110/ba-delij*cbd)
      c112=-2.0d0*c011/ba-c110
      c213=c011/(ba*ba)-c211
      c211=c211+c213
      c112=c112-5.0d0*c213/x2
      c110=c110+c112
      c011=c011-3.0d0*c112/x2
      go to 800
c  pc--p--db
  750 c011=-aijk*cbd
      c101t=baa(ka)/(3.0d0*ba)
      c101=c101t*aij
      c110= ak*(caa(ja)*ai+caa(ia)*aj)/(3.0d0*ca)
      c200=-0.333333333333333d0*(ai*deljk+aj*delik)
      c301=-0.6d0*(c200/ba-delij*c101t)
      c121=-c011/ca-c101
      c112=-2.0d0*c011/ba-c110
      c202=-2.0d0*c101/ba-c200
      c303=c101/(ba*ba)-c301
      c211=-0.6d0*(c110/ba+delij*ak*cbd)
      c213=c011/(ba*ba)-c211
      c220=-c110/ca-c200
      c321=-c211/ca-c301
      c222=-c112/ca-c202
      c323=-c213/ca-c303
      c222=c222-5.0d0*c323/x2
      c321=c321+c323
      c202=c202-5.0d0*c303/x2
      c202=c202+c222
      c112=c112-3.0d0*c222/x1
      c112=c112-5.0d0*c213/x2
      c301=c301+c303+c321
      c211=c211-3.0d0*c321/x1
      c211=c211+c213
      c200=c202+c220+c200
      c101=c101-3.0d0*c202/x2
```

```
      c101=c101+c121
      c110=c110-3.0d0*c220/x1
      c110=c110+c112
      c011=c011-3.0d0*c112/x2-3.0d0*c121/x1
      go to 800
c  dc--p--db
  760 c011=aijkl*cbd
      c110t=ai*caa(ja)+aj*caa(ia)
      c110=-akl*c110t/(3.0d0*ca)
      c101t=ak*baa(la)+al*baa(ka)
      c101=-aij*c101t/(3.0d0*ba)
      c112=-2.0d0*c011/ba-c110
      c121=-2.0d0*c011/ca-c101
      c211a=-0.6d0*(c110/ba-akl*delij*cbd)
      c211b=-0.6d0*(c101/ca-aij*delkl*cbd)
      c211=c211a+c211b
      c213=c011/(ba*ba)-c211a
      c231=c011/(ca*ca)-c211b
      c200=c2t*0.333333333333333d0
      c220=-2.0d0*c110/ca-c200
      c202=-2.0d0*c101/ba-c200
      c310=-(3.0d0*c200+delkl*c110t)/(5.0d0*ca)
      c301=-(3.0d0*c200+delij*c101t)/(5.0d0*ba)
      c330=c110/(ca*ca)-c310
      c303=c101/(ba*ba)-c301
      c222=-2.0d0*c112/ca-c202
      c321=-2.0d0*c211a/ca-c301
      c312=-2.0d0*c211b/ba-c310
      c323=-2.0d0*c213/ca-c303
      c332=-2.0d0*c231/ba-c330
      c411=-0.6d0*(c301/ca+c310/ba)-0.36d0*(c200/cba-delij*delkl*cbd)
      c431=c211a/(ca*ca)-c411
      c413=c211b/(ba*ba)-c411
      c433=c231/(ba*ba)-c431
      c431=c431+c433
      c332=c332-5.0d0*c433/x2
      c411=c411+c413+c431
      c312=c312-5.0d0*c413/x2
      c312=c312+c332
      c321=c321-5.0d0*c431/x1
      c222=c222-5.0d0*c332/x1
      c220=c220-5.0d0*c330/x1
      c222=c222-5.0d0*c323/x2
      c321=c321+c323
      c202=c202-5.0d0*c303/x2
      c202=c202+c222
      c112=c112-3.0d0*c222/x1
      c112=c112-5.0d0*c213/x2
      c121=c121-5.0d0*c231/x1
      c310=c310+c312+c330
      c301=c301+c303+c321
      c211=c211-3.0d0*c312/x2
      c211=c211-3.0d0*c321/x1
      c211=c211+c213+c231
      c200=c202+c220+c200
      c101=c101-3.0d0*c202/x2
      c101=c101+c121
      c110=c110-3.0d0*c220/x1
      c110=c110+c112
      c011=c011-3.0d0*c112/x2-3.0d0*c121/x1
      go to 800
c
  800 call vbcap(xcab)
c       write(60,4447?)xcab
```

```
c44472 format(1x,' xcab after <p|p> ',d15.9)
c
c
 1925 if(lmax(kc).lt.3)return
c
c      if(ddmx(kc) + d*(ca + ba)**2) 1950,1950,901
      buftest=d*(ca+ba)**2
      testcab=xcab
  901 ldstr =ldsr3(kc)
      ldstp =ldsp3(kc)
      a=zet1*2.0d0*ca
      b=zet2*2.0d0*ba
      x22=dsqrt(15.0d0)/2.0d0
      x21=2.0d0*x22
      x20=dsqrt(5.0d0)/2.0d0
      x2m1=x21
      x2m2=x21
      ax=caa(1)
      ay=caa(2)
      az=caa(3)
      bx=baa(1)
      by=baa(2)
      bz=baa(3)
      dax=ax/ca
      day=ay/ca
      daz=az/ca
      dbx=bx/ba
      dby=by/ba
      dbz=bz/ba
      bc=ba
      ac=ca
      daxs=dax*dax
      days=day*day
      dazs=daz*daz
      dbxs=dbx*dbx
      dbys=dby*dby
      dbzs=dbz*dbz
      cm211=0.0d0
      cm110=0.0d0
      cm101=0.0d0
      c000=0.0d0
      c011=0.0d0
      c101=0.0d0
      c110=0.0d0
      c200=0.0d0
      c211=0.0d0
      c301=0.0d0
      c310=0.0d0
      c400=0.0d0
      c411=0.0d0
      if (inttyp.ne.3)goto902
      ak=aj
      ka=ja
c     delik=delij
 902  bi=ai
      ib=ia
      jb=ja
      bj=aj
      dbi=bi/ba
      dbj=bj/ba
      dak=ak/ca
      dal=al/ca
      if(delij.ne.0.0d0)dij=1.0d0
      if(delkl.ne.0.0d0)dkl=1.0d0
```

```
      dxi=0.0d0
      dxj=0.0d0
      dxk=0.0d0
      dxl=0.0d0
      dyi=0.0d0
      dyj=0.0d0
      dyk=0.0d0
      dyl=0.0d0
      dzi=0.0d0
      dzj=0.0d0
      dzk=0.0d0
      dzl=0.0d0
      go to (905,910,915),ib
      go to 920
 905  dxi=1.0d0
      go to 920
 910  dyi=1.0d0
      go to 920
 915  dzi=1.0d0
 920  go to (925,930,935),jb
      go to 940
 925  dxj=1.0d0
      go to 940
 930  dyj=1.0d0
      go to 940
 935  dzj=1.0d0
 940  go to (945,950,955),ka
      go to 960
 945  dxk=1.0d0
      go to 960
 950  dyk=1.0d0
      go to 960
 955  dzk=1.0d0
 960  go to(965,970,975),la
      go to 980
 965  dxl=1.0d0
      go to 980
 970  dyl=1.0d0
      go to 980
 975  dzl=1.0d0
c
 980  go to (810,810,820,810,820,830),inttyp
c
 810  si00(1)=x22*(daxs-days)
      siml1(1)=-3.0d0*si00(1)/a
      si00(2)=x21*dax*daz
      siml1(2)=-3.0d0*si00(2)/a
      si00(3)=x20*(3.0d0*daz*daz-1.0d0)
      siml1(3)=-3.0d0*si00(3)/a
      si00(4)=x2m1*day*daz
      siml1(4)=-3.0d0*si00(4)/a
      si00(5)=x2m2*dax*day
      siml1(5)=-3.0d0*si00(5)/a
c       write(60,3180)
c3180   format(1x,' i         si00(i)         siml1(i)')
c       write(60,7321)(i,si00(i),siml1(i), i=1,5)
c 7321  format(1x,i3,d15.8,2x,d15.8)
c
      go to (840,850,7777,860),inttyp
c
 820  pi00(1)=x22*(ak*(days-daxs)-(5.0d0*dak*(daxs-days)+
     #       2.0d0*(day*dyk-dax*dxk))/a)
      piml1(1)=x22*(3.0d0*(ak*(daxs-days)+(5.0d0*dak*(daxs-
     #        days)+2.0d0*(day*dyk - dax*dxk))/a)/a)
```

```
      pi11(1)=x22*dak*(daxs-days)
      pi00(2)=x21*(-dax*daz*ak-(5.0d0*dax*daz*dak-dax*dzk
     #       -daz*dxk)/a)
      pim11(2)=x21*(3.0d0*(ak*dax*daz+(5.0d0*dax*daz*dak-
     #        dax*dzk - daz*dxk)/a)/a)
      pi11(2)=dax*daz*dak*x21
      pi00(3)=x20*(ak*(1.0d0-3.0d0*dazs)-3.0d0*(5.0d0*
     #       dak*dazs - 2.0d0*daz*dzk -dak)/a)
      pim11(3)=x20*(3.0d0*(ak*(3.0d0*dazs-1.0d0)+3.0d0*
     #        (5.0d0*dak*dazs - 2.0d0*daz*dzk - dak)/a)/a)
      pi11(3)=x20*(3.0d0*dak*dazs-dak)
      pi00(4)=x2m1*(-daz*day*ak-(5.0d0*day*daz*dak-day*dzk
     #       -daz*dyk)/a)
      pim11(4)=x2m1*(3.0d0*(ak*day*daz+(5.0d0*day*daz*dak-
     #        day*dzk - daz*dyk)/a)/a)
      pi11(4)=day*daz*dak*x2m1
      pi00(5)=x2m2*(-ak*dax*day-(5.0d0*dax*day*dak-dax*dyk
     #       -day*dxk)/a)
      pim11(5)=x2m2*(3.0d0*(ak*dax*day+(5.0d0*dax*day*dak-
     #        dax*dyk - day*dxk)/a)/a)
      pi11(5)=dax*day*dak*x2m2
c
cpi00(1)=- x22*( ak*ac*ax**2.d0*a - ak*ac*ay**2.d0*a +
c     #       5.0d0*ak*ax**2.d0 - 5.0d0*ak*ay**2.d0 -
c     #       2.d0*ac**2.d0*ax*dxk + 2.d0*ac**2.d0*ay*dyk )
c     #       / ( ac**3.0d0*a )
c      pim11(1)=3.0d0 *x22* ( ak*ac*ax**2.d0*a - ak*ac*ay**2.d0*a +
c     #       5.0d0*ak*ax**2.d0 - 5.0d0*ak*ay**2.d0 -
c     #       2.d0*ac**2.d0*ax*dxk + 2.d0*ac**2.d0*ay*dyk ) /
c     #       ( ac**3.0d0*a**2.d0 )
c      pi11(1)=ak *x22* ( ax**2.d0 - ay**2.d0 ) / ( ac**3.0d0 )
c      pi00(2)=- x21*( ak*ac*az*ax*a + 5.0d0*ak*az*ax -
c     #         ac**2.d0*az*dxk - ac**2.d0*dzk*ax ) /
c     #         ( ac**3.0d0*a )
c      pim11(2)=3.0d0 *x21* ( ak*ac*az*ax*a + 5.0d0*ak*az*ax -
c     #        ac**2.d0*az*dxk - ac**2.d0*dzk*ax ) /
c     #        ( ac**3.0d0*a**2.d0 )
cc      pi11(2)=x21*ak*az*ax / ( ac**3.0d0 )
c      pi00(3)=x20*( ak*ac**3.0d0*a + 3.0d0*ak*ac**2.d0 -
c     #       3.0d0*ak*ac*az**2.d0*a - 15.0d0*ak*az**2.d0 +
c     #      6.0d0*ac**2.d0*az*dzk ) / ( ac**3.0d0*a )
c     pim11(3)=-x20* 3.0d0 * ( ak*ac**3.0d0*a + 3.0d0*ak*ac**2.d0 -
c     #       3.0d0*ak*ac*az**2.d0*a - 15.0d0*ak*az**2.d0 +
c     #       6.0d0*ac**2.d0*az*dzk ) / ( ac**3.0d0*a**2.d0 )
cc     pi11(3)=- x20*ak * ( ac**2.d0 - 3.0d0*az**2.d0 ) / ( ac**3.0d0)
c     pi00(4)=- x2m1*(ak*ac*az*ay*a+5.0d0*ak*az*ay-ac**2.d0*az*dyk-
c     #        ac**2.d0*dzk*ay ) / ( ac**3.0d0*a )
c      pim11(4)=3.0d0 *x2m1* ( ak*ac*az*ay*a + 5.0d0*ak*az*ay -
c     #        ac**2.d0*az*dyk - ac**2.d0*dzk*ay ) /
c     #        ( ac**3.0d0*a**2.d0 )
c      pi11(4)=x2m1*ak*az*ay / ( ac**3.0d0 )
c      pi00(5)=-x2m2*(ak*ac*ax*ay*a+5.0d0*ak*ax*ay-ac**2.d0*ax*dyk-
c     #       ac**2.d0*dxk*ay ) / ( ac**3.0d0*a )
c      pim11(5)=3.0d0 *x2m2* ( ak*ac*ax*ay*a + 5.0d0*ak*ax*ay -
c     #       ac**2.d0*ax*dyk - ac**2.d0*dxk*ay ) /
c     #       ( ac**3.0d0*a**2.d0 )
c      pi11(5)=x2m2*ak*ax*ay / ( ac**3.0d0 )
c
      go to (850,7777,860),(inttyp-2)
c
c    d type
c
c@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
c
```

```
c   z22
c# di00
c
 830  di00(1)=x22*(ak*al*ac**2*a**2*ax**2-ak*al*ac**2*a**2*ay**2+
     # 10.0d0*ak*al*ac*a*ax**2-10.0d0*ak*al*ac*a*ay**2+35.0d0*ak*
     # al*ax**2 - 35.d0*ak*al*ay**2 -
     # 2.d0*ak*ac**3*a*ax*dxl + 2.d0*ak*ac**3*a*ay*dyl -
     # 10.0d0*ak*ac**2*ax*dxl + 10.0d0*ak*ac**2*ay*dyl -
     # 2.d0*al*ac**3*a*ax*dxk + 2.d0*al*ac**3*a*ay*dyk -
     # 10.0d0*al*ac**2*ax*dxk + 10.0d0*al*ac**2*ay*dyk +
     # 2.d0*ac**4*dxk*dxl - 2.d0*ac**4*dyk*dyl-2.d0*ac**3*dkl*a*ax**2 +
     # 2.d0*ac**3*dkl*a*ay**2 - 5.d0*ac**2*dkl*ax**2 +
     # 5.d0*ac**2*dkl*ay**2 ) / ( ac**4*a**2 )
c
c dim11
c
      dim11(1)=-x22*(3.d0*ak*al*ac**2*a**2*ax**2-
     # 3.d0*ak*al*ac**2*a**2*ay**2 + 30.0d0*ak*al*ac*a*ax**2 -
     # 30.0d0*ak*al*ac*a*ay**2 + 105.d0*ak*al*ax**2 -
     # 105.d0*ak*al*ay**2 - 6.d0*ak*ac**3*a*ax*dxl +
     # 6.d0*ak*ac**3*a*ay*dyl - 30.0d0*ak*ac**2*ax*dxl +
     # 30.0d0*ak*ac**2*ay*dyl - 6.d0*al*ac**3*a*ax*dxk +
     # 6.d0*al*ac**3*a*ay*dyk - 30.0d0*al*ac**2*ax*dxk +
     # 30.0d0*al*ac**2*ay*dyk - ac**4*dkl*a**2*ax**2 +
     # ac**4*dkl*a**2*ay**2 + 6.d0*ac**4*dxk*dxl -
     # 6.d0*ac**4*dyk*dyl - 6.d0*ac**3*dkl*a*ax**2 +
     # 6.d0*ac**3*dkl*a*ay**2 - 15.d0*ac**2*dkl*ax**2 +
     # 15.d0*ac**2*dkl*ay**2 ) / ( ac**4*a**3 )
c
c  di11
c
      di11(1)=-x22*(2.d0*ak*al*ac*a*ax**2-2.d0*ak*al*ac*a*ay**2+
     # 10.0d0*ak*al*ax**2 - 10.0d0*ak*al*ay**2 - 2.d0*ak*ac**2*ax*dxl +
     # 2.d0*ak*ac**2*ay*dyl - 2.d0*al*ac**2*ax*dxk +
     # 2.d0*al*ac**2*ay*dyk - ac**2*dkl*ax**2 + ac**2*dkl*ay**2 ) /
     # ( ac**4*a )
c
c  di20
c
      di20(1)=x22*ak*al*(ax**2-ay**2)/(ac**4)
c
c
cz21
c  di00
c
      di00(2)=x21*(ak*al*ac**2*a**2*ax*az+10.0d0*ak*al*ac*a*ax*az+
     # 35.d0*ak*al*ax*az - ak*ac**3*a*ax*dzl - ak*ac**3*a*az*dxl -
     # 5.d0*ak*ac**2*ax*dzl - 5.d0*ak*ac**2*az*dxl - al*ac**3*a*ax*dzk -
     # al*ac**3*a*az*dxk - 5.d0*al*ac**2*ax*dzk - 5.d0*al*ac**2*az*dxk +
     # ac**4*dxk*dzl + ac**4*dxl*dzk - 2.d0*ac**3*dkl*a*ax*az -
     # 5.d0*ac**2*dkl*ax*az ) / ( ac**4*a**2 )
c
c  dim11
c
      dim11(2)=-x21*(3.d0*ak*al*ac**2*a**2*ax*az+
     # 30.0d0*ak*al*ac*a*ax*az + 105.d0*ak*al*ax*az -
     # 3.d0*ak*ac**3*a*ax*dzl - 3.d0*ak*ac**3*a*az*dxl -
     # 15.d0*ak*ac**2*ax*dzl - 15.d0*ak*ac**2*az*dxl -
     # 3.d0*al*ac**3*a*ax*dzk - 3.d0*al*ac**3*a*az*dxk -
     # 15.d0*al*ac**2*ax*dzk - 15.d0*al*ac**2*az*dxk -
     # ac**4*dkl*a**2*ax*az + 3.d0*ac**4*dxk*dzl + 3.d0*ac**4*dxl*dzk -
     # 6.d0*ac**3*dkl*a*ax*az - 15.d0*ac**2*dkl*ax*az )/( ac**4*a**3 )
c
c  di11
```

```
c
      dil1(2)=-x21*(2.d0*ak*al*ac*a*ax*az+10.0d0*ak*al*ax*az-
     # ak*ac**2*ax*dzl - ak*ac**2*az*dxl - al*ac**2*ax*dzk -
     # al*ac**2*az*dxk - ac**2*dkl*ax*az ) / ( ac**4*a )
c
c
c  di20
c
      di20(2)=x21*ak*al*ax*az/(ac**4)
c
c  d20
c  dk00
c
      di00(3)=-x20*(ak*al*ac**2*a**2*ax**2+ak*al*ac**2*a**2*ay**2-
     # 2.d0*ak*al*ac**2*a**2*az**2 + 10.0d0*ak*al*ac*a*ax**2 +
     # 10.0d0*ak*al*ac*a*ay**2 - 20.0d0*ak*al*ac*a*az**2 +
     # 35.d0*ak*al*ax**2 +  35.d0*ak*al*ay**2 - 70.0d0*ak*al*az**2 -
     # 2.d0*ak*ac**3*a*ax*dxl - 2.d0*ak*ac**3*a*ay*dyl +
     # 4.d0*ak*ac**3*a*az*dzl - 10.0d0*ak*ac**2*ax*dxl -
     # 10.0d0*ak*ac**2*ay*dyl + 20.0d0*ak*ac**2*az*dzl -
     # 2.d0*al*ac**3*a*ax*dxk - 2.d0*al*ac**3*a*ay*dyk +
     # 4.d0*al*ac**3*a*az*dzk - 10.0d0*al*ac**2*ax*dxk -
     # 10.0d0*al*ac**2*ay*dyk + 20.0d0*al*ac**2*az*dzk +
     # 2.d0*ac**4*dxk*dxl + 2.d0*ac**4*dyk*dyl - 4.d0*ac**4*dzk*dzl -
     # 2.d0*ac**3*dkl*a*ax**2 - 2.d0*ac**3*dkl*a*ay**2 +
     # 4.d0*ac**3*dkl*a*az**2 - 5.d0*ac**2*dkl*ax**2 -
     # 5.d0*ac**2*dkl*ay**2 + 10.0d0*ac**2*dkl*az**2 ) / ( ac**4*a**2 )
c
c dkm11
c
      dim11(3)=x20*(3.d0*ak*al*ac**2*a**2*ax**2+
     # 3.d0*ak*al*ac**2*a**2*ay**2 - 6.d0*ak*al*ac**2*a**2*az**2 +
     # 30.0d0*ak*al*ac*a*ax**2 + 30.0d0*ak*al*ac*a*ay**2 -
     # 60.0d0*ak*al*ac*a*az**2 + 105.d0*ak*al*ax**2 +
     # 105.d0*ak*al*ay**2 - 210.0d0*ak*al*az**2 -
     # 6.d0*ak*ac**3*a*ax*dxl - 6.d0*ak*ac**3*a*ay*dyl +
     # 12.d0*ak*ac**3*a*az*dzl - 30.0d0*ak*ac**2*ax*dxl -
     # 30.0d0*ak*ac**2*ay*dyl + 60.0d0*ak*ac**2*az*dzl -
     # 6.d0*al*ac**3*a*ax*dxk - 6.d0*al*ac**3*a*ay*dyk +
     # 12.d0*al*ac**3*a*az*dzk - 30.0d0*al*ac**2*ax*dxk -
     # 30.0d0*al*ac**2*ay*dyk + 60.0d0*al*ac**2*az*dzk -
     # ac**4*dkl*a**2*ax**2 - ac**4*dkl*a**2*ay**2 +
     # 2.d0*ac**4*dkl*a**2*az**2 + 6.d0*ac**4*dxk*dxl +
     # 6.d0*ac**4*dyk*dyl - 12.d0*ac**4*dzk*dzl -
     # 6.d0*ac**3*dkl*a*ax**2 - 6.d0*ac**3*dkl*a*ay**2 +
     # 12.d0*ac**3*dkl*a*az**2 - 15.d0*ac**2*dkl*ax**2 -
     # 15.d0*ac**2*dkl*ay**2 + 30.0d0*ac**2*dkl*az**2 ) /
     # ( ac**4*a**3 )
c
c
c# di11
c
      di11(3)=x20*(2.d0*ak*al*ac*a*ax**2+2.d0*ak*al*ac*a*ay**2-
     # 4.d0*ak*al*ac*a*az**2 + 10.0d0*ak*al*ax**2 +
     # 10.0d0*ak*al*ay**2 - 20.0d0*ak*al*az**2 -
     # 2.d0*ak*ac**2*ax*dxl - 2.d0*ak*ac**2*ay*dyl +
     # 4.d0*ak*ac**2*az*dzl - 2.d0*al*ac**2*ax*dxk -
     # 2.d0*al*ac**2*ay*dyk + 4.d0*al*ac**2*az*dzk - ac**2*dkl*ax**2 -
     # ac**2*dkl*ay**2 + 2.d0*ac**2*dkl*az**2 ) / ( ac**4*a )
c
c# di20
c
      di20(3)=-x20*ak*al*(ax**2+ay**2-2.d0*az**2)/(ac**4)
c
```

```
c  d2ml
c# di00
c
c
      di00(4)=x2ml*(ak*al*ac**2*a**2*az*ay+10.0d0*ak*al*ac*a*az*ay+
     # 35.d0*ak*al*az*ay - ak*ac**3*a*az*dyl - ak*ac**3*a*ay*dzl -
     # 5.d0*ak*ac**2*az*dyl - 5.d0*ak*ac**2*ay*dzl -
     # al*ac**3*a*az*dyk - al*ac**3*a*ay*dzk - 5.d0*al*ac**2*az*dyk -
     # 5.d0*al*ac**2*ay*dzk + ac**4*dzk*dyl + ac**4*dzl*dyk -
     # 2.d0*ac**3*dkl*a*az*ay - 5.d0*ac**2*dkl*az*ay )/( ac**4*a**2 )
c
c# diml1
c
      diml1(4)=-x2ml*(3.d0*ak*al*ac**2*a**2*az*ay+
     # 30.0d0*ak*al*ac*a*az*ay + 105.d0*ak*al*az*ay -
     # 3.d0*ak*ac**3*a*az*dyl - 3.d0*ak*ac**3*a*ay*dzl -
     # 15.d0*ak*ac**2*az*dyl - 15.d0*ak*ac**2*ay*dzl -
     # 3.d0*al*ac**3*a*az*dyk - 3.d0*al*ac**3*a*ay*dzk -
     # 15.d0*al*ac**2*az*dyk - 15.d0*al*ac**2*ay*dzk -
     # ac**4*dkl*a**2*az*ay + 3.d0*ac**4*dzk*dyl + 3.d0*ac**4*dzl*dyk -
     # 6*ac**3*dkl*a*az*ay - 15.d0*ac**2*dkl*az*ay ) / ( ac**4*a**3 )
c
c
c# di11
c
      di11(4)=-x2ml*(2.d0*ak*al*ac*a*az*ay+10.0d0*ak*al*az*ay-
     # ak*ac**2*az*dyl - ak*ac**2*ay*dzl - al*ac**2*az*dyk -
     # al*ac**2*ay*dzk - ac**2*dkl*az*ay ) / ( ac**4*a )
c
c# di20
c
      di20(4)=x2ml*ak*al*az*ay/(ac**4)
c
c
cd2m2
c# di00
c
c
      di00(5)=x2m2*(ak*al*ac**2*a**2*ax*ay+10.0d0*ak*al*ac*a*ax*ay+
     # 35.d0*ak*al*ax*ay - ak*ac**3*a*ax*dyl - ak*ac**3*a*ay*dxl -
     # 5.d0*ak*ac**2*ax*dyl - 5.d0*ak*ac**2*ay*dxl -
     # al*ac**3*a*ax*dyk - al*ac**3*a*ay*dxk - 5.d0*al*ac**2*ax*dyk -
     # 5.d0*al*ac**2*ay*dxk + ac**4*dxk*dyl + ac**4*dxl*dyk -
     # 2.d0*ac**3*dkl*a*ax*ay - 5.d0*ac**2*dkl*ax*ay ) / ( ac**4*a**2 )
c
c# diml1
c
      diml1(5)=-x2m2*(3.d0*ak*al*ac**2*a**2*ax*ay+
     # 30.0d0*ak*al*ac*a*ax*ay + 105.d0*ak*al*ax*ay -
     # 3.d0*ak*ac**3*a*ax*dyl - 3.d0*ak*ac**3*a*ay*dxl -
     # 15.d0*ak*ac**2*ax*dyl - 15.d0*ak*ac**2*ay*dxl -
     # 3.d0*al*ac**3*a*ax*dyk - 3.d0*al*ac**3*a*ay*dxk -
     # 15.d0*al*ac**2*ax*dyk - 15.d0*al*ac**2*ay*dxk -
     # ac**4*dkl*a**2*ax*ay + 3.d0*ac**4*dxk*dyl +
     # 3.d0*ac**4*dxl*dyk - 6.d0*ac**3*dkl*a*ax*ay -
     # 15.d0*ac**2*dkl*ax*ay ) / ( ac**4*a**3 )
c
c# di11
c
      di11(5)=-x2m2*(2.d0*ak*al*ac*a*ax*ay+10.0d0*ak*al*ax*ay-
     # ak*ac**2*ax*dyl - ak*ac**2*ay*dxl - al*ac**2*ax*dyk -
     # al*ac**2*ay*dxk - ac**2*dkl*ax*ay ) / ( ac**4*a )
c
c# di20
```

```
c
      di20(5)=x2m2*ak*al*ax*ay/(ac**4)
c
c
c
c@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
c
      go to 860
 840  sj00(1)=x22*(dbxs-dbys)
      sjm11(1)=-3.0d0*sj00(1)/b
      sj00(2)=x21*dbx*dbz
      sjm11(2)=-3.0d0*sj00(2)/b
      sj00(3)=x20*(3.0d0*dbz*dbz-1.0d0)
      sjm11(3)=-3.0d0*sj00(3)/b
      sj00(4)=x2m1*dby*dbz
      sjm11(4)=-3.0d0*sj00(4)/b
      sj00(5)=x2m2*dbx*dby
      sjm11(5)=-3.0d0*sj00(5)/b
c
      go to 890
c
 850  pj00(1)=x22*(bi*(dbys-dbxs)-(5.0d0*dbi*(dbxs-dbys)+
     #        2.0d0*(dby*dyi-dbx*dxi))/b)
      pjm11(1)=x22*(3.0d0*(bi*(dbxs-dbys)+(5.0d0*dbi*(dbxs
     #       -dbys)+2.0d0*(dby*dyi - dbx*dxi))/b)/b)
      pj11(1)=dbi*(dbxs-dbys)*x22
      pj00(2)=x21*(-dbx*dbz*bi-(5.0d0*dbx*dbz*dbi-dbx*dzi
     #       -dbz*dxi)/b)
      pjm11(2)=x21*(3.0d0*(bi*dbx*dbz+(5.0d0*dbx*dbz*dbi-
     #        dbx*dzi - dbz*dxi)/b)/b)
      pj11(2)=dbx*dbz*dbi*x21
      pj00(3)=x20*(bi*(1.0d0-3.0d0*dbzs)-3.0d0*(5.0d0*
     #       dbi*dbzs - 2.0d0*dbz*dzi -dbi)/b)
      pjm11(3)=x20*(3.0d0*(bi*(3.0d0*dbzs-1.0d0)+3.0d0*
     #        (5.0d0*dbi*dbzs - 2.0d0*dbz*dzi - dbi)/b)/b)
      pj11(3)=x20*(3.0d0*dbi*dbzs-dbi)
      pj00(4)=x2m1*(-dbz*dby*bi-(5.0d0*dby*dbz*dbi-dby*dzi
     #       -dbz*dyi)/b)
      pjm11(4)=x2m1*(3.0d0*(bi*dby*dbz+(5.0d0*dby*dbz*dbi-
     #        dby*dzi - dbz*dyi)/b)/b)
      pj11(4)=dby*dbz*dbi*x2m1
      pj00(5)=x2m2*(-bi*dbx*dby-(5.0d0*dbx*dby*dbi-dbx*dyi
     #       -dby*dxi)/b)
      pjm11(5)=x2m2*(3.0d0*(bi*dbx*dby+(5.0d0*dbx*dby*dbi-
     #        dbx*dyi - dby*dxi)/b)/b)
      pj11(5)=dbx*dby*dbi*x2m2
cpj00(1)=- x22*( bi*bc*bx**2.d0*b - bi*bc*by**2.d0*b +
c    #       5.0d0*bi*bx**2.d0 - 5.0d0*bi*by**2.d0 -
c    #       2.d0*bc**2.d0*bx*dxi + 2.d0*bc**2.d0*by*dyi )
c    #       / ( bc**3.0d0*b )
c      pjm11(1)=3.0d0 *x22* ( bi*bc*bx**2.d0*b - bi*bc*by**2.d0*b +
c    #       5.0d0*bi*bx**2.d0 - 5.0d0*bi*by**2.d0 -
c    #       2.d0*bc**2.d0*bx*dxi + 2.d0*bc**2.d0*by*dyi ) /
c    #       ( bc**3.0d0*b**2.d0 )
c      pj11(1)=bi *x22* ( bx**2.d0 - by**2.d0 ) / ( bc**3.0d0 )
c      pj00(2)=- x21*( bi*bc*bz*bx*b + 5.0d0*bi*bz*bx -
c    #         bc**2.d0*bz*dxi - bc**2.d0*dzi*bx ) /
c    #         ( bc**3.0d0*b )
c      pjm11(2)=3.0d0 *x21* ( bi*bc*bz*bx*b + 5.0d0*bi*bz*bx -
c    #        bc**2.d0*bz*dxi - bc**2.d0*dzi*bx ) /
c    #        ( bc**3.0d0*b**2.d0 )
c      pj11(2)=x21*bi*bz*bx / ( bc**3.0d0 )
c      pj00(3)=x20*( bi*bc**3.0d0*b + 3.0d0*bi*bc**2.d0 -
c    #       3.0d0*bi*bc*bz**2.d0*b - 15.0d0*bi*bz**2.d0 +
```

```
c     #      6.0d0*bc**2.d0*bz*dzi ) / ( bc**3.0d0*b )
c      pjml1(3)=-x20* 3.0d0 * ( bi*bc**3.0d0*b + 3.0d0*bi*bc**2.d0 -
c     #       3.0d0*bi*bc*bz**2.d0*b - 15.0d0*bi*bz**2.d0 +
c     #       6.0d0*bc**2.d0*bz*dzi ) / ( bc**3.0d0*b**2.d0 )
c      pj11(3)=- x20*bi * ( bc**2.d0 - 3.0d0*bz**2.d0 ) / ( bc**3.0d0 )
c      pj00(4)=-x2m1*(bi*bc*bz*by*b+5.0d0*bi*bz*by - bc**2.d0*bz*dyi -
c     #       bc**2.d0*dzi*by ) /( bc**3.0d0*b )
c      pjml1(4)=3.0d0 *x2m1* ( bi*bc*bz*by*b + 5.0d0*bi*bz*by -
c     #       bc**2.d0*bz*dyi - bc**2.d0*dzi*by ) /
c     #       ( bc**3.0d0*b**2.d0 )
c      pj11(4)=x2m1*bi*bz*by / ( bc**3.0d0 )
c      pj00(5)=-x2m2*(bi*bc*bx*by*b+5.0d0*bi*bx*by-bc**2.d0*bx*dyi-
c     #      bc**2.d0*dxi*by ) / ( bc**3.0d0*b )
c      pjml1(5)=3.0d0 *x2m2* ( bi*bc*bx*by*b + 5.0d0*bi*bx*by -
c     #      bc**2.d0*bx*dyi - bc**2.d0*dxi*by ) /
c     #      ( bc**3.0d0*b**2.d0 )
c      pj11(5)=x2m2*bi*bx*by / ( bc**3.0d0 )
c
c      write(60,3256)
c3256  format(1x,' i       pj00(i)       pjml1(i)       pj11(i)')
c      write(60,8321)(i,pj00(i),pjml1(i),pj11(i), i=1,5)
c 8321 format(1x,i3,d15.8,2x,d15.8,2x,d15.8)
c
      go to 890
c
c  d type integrals
c
c@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
c
c
c   z22
c# di00
c
 860  dj00(1)=x22*(bi*bj*bc**2*b**2*bx**2-bi*bj*bc**2*b**2*by**2+
     # 10.0d0*bi*bj*bc*b*bx**2-10.0d0*bi*bj*bc*b*by**2+35.0d0*bi*
     # bj*bx**2 - 35.d0*bi*bj*by**2 -
     # 2.d0*bi*bc**3*b*bx*dxj + 2.d0*bi*bc**3*b*by*dyj -
     # 10.0d0*bi*bc**2*bx*dxj + 10.0d0*bi*bc**2*by*dyj -
     # 2.d0*bj*bc**3*b*bx*dxi + 2.d0*bj*bc**3*b*by*dyi -
     # 10.0d0*bj*bc**2*bx*dxi + 10.0d0*bj*bc**2*by*dyi +
     # 2.d0*bc**4*dxi*dxj - 2.d0*bc**4*dyi*dyj-2.d0*bc**3*dij*b*bx**2 +
     # 2.d0*bc**3*dij*b*by**2 - 5.d0*bc**2*dij*bx**2 +
     # 5.d0*bc**2*dij*by**2 ) / ( bc**4*b**2 )
c
c# diml1
c
      djml1(1)=-x22*(3.d0*bi*bj*bc**2*b**2*bx**2-
     # 3.d0*bi*bj*bc**2*b**2*by**2 + 30.0d0*bi*bj*bc*b*bx**2 -
     # 30.0d0*bi*bj*bc*b*by**2 + 105.d0*bi*bj*bx**2 -
     # 105.d0*bi*bj*by**2 - 6.d0*bi*bc**3*b*bx*dxj +
     # 6.d0*bi*bc**3*b*by*dyj - 30.0d0*bi*bc**2*bx*dxj +
     # 30.0d0*bi*bc**2*by*dyj - 6.d0*bj*bc**3*b*bx*dxi +
     # 6.d0*bj*bc**3*b*by*dyi - 30.0d0*bj*bc**2*bx*dxi +
     # 30.0d0*bj*bc**2*by*dyi - bc**4*dij*b**2*bx**2 +
     # bc**4*dij*b**2*by**2 + 6.d0*bc**4*dxi*dxj -
     # 6.d0*bc**4*dyi*dyj - 6.d0*bc**3*dij*b*bx**2 +
     # 6.d0*bc**3*dij*b*by**2 - 15.d0*bc**2*dij*bx**2 +
     # 15.d0*bc**2*dij*by**2 ) / ( bc**4*b**3 )
c
c# di11
c
      dj11(1)=-x22*(2.d0*bi*bj*bc*b*bx**2-2.d0*bi*bj*bc*b*by**2+
     # 10.0d0*bi*bj*bx**2 - 10.0d0*bi*bj*by**2 - 2.d0*bi*bc**2*bx*dxj +
     # 2.d0*bi*bc**2*by*dyj - 2.d0*bj*bc**2*bx*dxi +
```

```
     # 2.d0*bj*bc**2*by*dyi - bc**2*dij*bx**2 + bc**2*dij*by**2 ) /
     # ( bc**4*b )
c
c# di20
c
      dj20(1)=x22*bi*bj*(bx**2-by**2)/(bc**4)
c
c
cz21
c# di00
c
      dj00(2)=x21*(bi*bj*bc**2*b**2*bx*bz+10.0d0*bi*bj*bc*b*bx*bz+
     # 35.d0*bi*bj*bx*bz - bi*bc**3*b*bx*dzj - bi*bc**3*b*bz*dxj -
     # 5.d0*bi*bc**2*bx*dzj - 5.d0*bi*bc**2*bz*dxj - bj*bc**3*b*bx*dzi -
     # bj*bc**3*b*bz*dxi - 5.d0*bj*bc**2*bx*dzi - 5.d0*bj*bc**2*bz*dxi +
     # bc**4*dxi*dzj + bc**4*dxj*dzi - 2.d0*bc**3*dij*b*bx*bz -
     # 5.d0*bc**2*dij*bx*bz ) / ( bc**4*b**2 )
c
c# dim11
c
      djm11(2)=-x21*(3.d0*bi*bj*bc**2*b**2*bx*bz+
     # 30.0d0*bi*bj*bc*b*bx*bz + 105.d0*bi*bj*bx*bz -
     # 3.d0*bi*bc**3*b*bx*dzj - 3.d0*bi*bc**3*b*bz*dxj -
     # 15.d0*bi*bc**2*bx*dzj - 15.d0*bi*bc**2*bz*dxj -
     # 3.d0*bj*bc**3*b*bx*dzi - 3.d0*bj*bc**3*b*bz*dxi -
     # 15.d0*bj*bc**2*bx*dzi - 15.d0*bj*bc**2*bz*dxi -
     # bc**4*dij*b**2*bx*bz + 3.d0*bc**4*dxi*dzj + 3.d0*bc**4*dxj*dzi -
     # 6.d0*bc**3*dij*b*bx*bz - 15.d0*bc**2*dij*bx*bz )/( bc**4*b**3 )
c
c# di11
c
      dj11(2)=-x21*(2.d0*bi*bj*bc*b*bx*bz+10.0d0*bi*bj*bx*bz-
     # bi*bc**2*bx*dzj - bi*bc**2*bz*dxj - bj*bc**2*bx*dzi -
     # bj*bc**2*bz*dxi - bc**2*dij*bx*bz ) / ( bc**4*b )
c
c
c# di20
c
      dj20(2)=x21*bi*bj*bx*bz/(bc**4)
c
c  d20
c di00
c
      dj00(3)=-x20*(bi*bj*bc**2*b**2*bx**2+bi*bj*bc**2*b**2*by**2-
     # 2.d0*bi*bj*bc**2*b**2*bz**2 + 10.0d0*bi*bj*bc*b*bx**2 +
     # 10.0d0*bi*bj*bc*b*by**2 - 20.0d0*bi*bj*bc*b*bz**2 +
     # 35.d0*bi*bj*bx**2 +  35.d0*bi*bj*by**2 - 70.0d0*bi*bj*bz**2 -
     # 2.d0*bi*bc**3*b*bx*dxj - 2.d0*bi*bc**3*b*by*dyj +
     # 4.d0*bi*bc**3*b*bz*dzj - 10.0d0*bi*bc**2*bx*dxj -
     # 10.0d0*bi*bc**2*by*dyj + 20.0d0*bi*bc**2*bz*dzj -
     # 2.d0*bj*bc**3*b*bx*dxi - 2.d0*bj*bc**3*b*by*dyi +
     # 4.d0*bj*bc**3*b*bz*dzi - 10.0d0*bj*bc**2*bx*dxi -
     # 10.0d0*bj*bc**2*by*dyi + 20.0d0*bj*bc**2*bz*dzi +
     # 2.d0*bc**4*dxi*dxj + 2.d0*bc**4*dyi*dyj - 4.d0*bc**4*dzi*dzj -
     # 2.d0*bc**3*dij*b*bx**2 - 2.d0*bc**3*dij*b*by**2 +
     # 4.d0*bc**3*dij*b*bz**2 - 5.d0*bc**2*dij*bx**2 -
     # 5.d0*bc**2*dij*by**2 + 10.0d0*bc**2*dij*bz**2 ) / ( bc**4*b**2 )
c
c dim11
c
      djm11(3)=x20*(3.d0*bi*bj*bc**2*b**2*bx**2+
     # 3.d0*bi*bj*bc**2*b**2*by**2 - 6.d0*bi*bj*bc**2*b**2*bz**2 +
     # 30.0d0*bi*bj*bc*b*bx**2 + 30.0d0*bi*bj*bc*b*by**2 -
     # 60.0d0*bi*bj*bc*b*bz**2 + 105.d0*bi*bj*bx**2 +
```

```
     # 105.d0*bi*bj*by**2 - 210.0d0*bi*bj*bz**2 -
     # 6.d0*bi*bc**3*b*bx*dxj - 6.d0*bi*bc**3*b*by*dyj +
     # 12.d0*bi*bc**3*b*bz*dzj - 30.0d0*bi*bc**2*bx*dxj -
     # 30.0d0*bi*bc**2*by*dyj + 60.0d0*bi*bc**2*bz*dzj -
     # 6.d0*bj*bc**3*b*bx*dxi - 6.d0*bj*bc**3*b*by*dyi +
     # 12.d0*bj*bc**3*b*bz*dzi - 30.0d0*bj*bc**2*bx*dxi -
     # 30.0d0*bj*bc**2*by*dyi + 60.0d0*bj*bc**2*bz*dzi -
     # bc**4*dij*b**2*bx**2 - bc**4*dij*b**2*by**2 +
     # 2.d0*bc**4*dij*b**2*bz**2 + 6.d0*bc**4*dxi*dxj +
     # 6.d0*bc**4*dyi*dyj - 12.d0*bc**4*dzi*dzj -
     # 6.d0*bc**3*dij*b*bx**2 - 6.d0*bc**3*dij*b*by**2 +
     # 12.d0*bc**3*dij*b*bz**2 - 15.d0*bc**2*dij*bx**2 -
     # 15.d0*bc**2*dij*by**2 + 30.0d0*bc**2*dij*bz**2 ) /
     # ( bc**4*b**3 )
c
c
c# di11
c
      dj11(3)=x20*(2.d0*bi*bj*bc*b*bx**2+2.d0*bi*bj*bc*b*by**2-
     # 4.d0*bi*bj*bc*b*bz**2 + 10.0d0*bi*bj*bx**2 +
     # 10.0d0*bi*bj*by**2 - 20.0d0*bi*bj*bz**2 -
     # 2.d0*bi*bc**2*bx*dxj - 2.d0*bi*bc**2*by*dyj +
     # 4.d0*bi*bc**2*bz*dzj - 2.d0*bj*bc**2*bx*dxi -
     # 2.d0*bj*bc**2*by*dyi + 4.d0*bj*bc**2*bz*dzi - bc**2*dij*bx**2 -
     # bc**2*dij*by**2 + 2.d0*bc**2*dij*bz**2 ) / ( bc**4*b )
c
c# di20
c
      dj20(3)=-x20*bi*bj*(bx**2+by**2-2.d0*bz**2)/(bc**4)
c
c  d2m1
c# di00
c
c
      dj00(4)=x2m1*(bi*bj*bc**2*b**2*bz*by+10.0d0*bi*bj*bc*b*bz*by+
     # 35.d0*bi*bj*bz*by - bi*bc**3*b*bz*dyj - bi*bc**3*b*by*dzj -
     # 5.d0*bi*bc**2*bz*dyj - 5.d0*bi*bc**2*by*dzj -
     # bj*bc**3*b*bz*dyi - bj*bc**3*b*by*dzi - 5.d0*bj*bc**2*bz*dyi -
     # 5.d0*bj*bc**2*by*dzi + bc**4*dzi*dyj + bc**4*dzj*dyi -
     # 2.d0*bc**3*dij*b*bz*by - 5.d0*bc**2*dij*bz*by )/( bc**4*b**2 )
c
c# dim11
c
      djm11(4)=-x2m1*(3.d0*bi*bj*bc**2*b**2*bz*by+
     # 30.0d0*bi*bj*bc*b*bz*by + 105.d0*bi*bj*bz*by -
     # 3.d0*bi*bc**3*b*bz*dyj - 3.d0*bi*bc**3*b*by*dzj -
     # 15.d0*bi*bc**2*bz*dyj - 15.d0*bi*bc**2*by*dzj -
     # 3.d0*bj*bc**3*b*bz*dyi - 3.d0*bj*bc**3*b*by*dzi -
     # 15.d0*bj*bc**2*bz*dyi - 15.d0*bj*bc**2*by*dzi -
     # bc**4*dij*b**2*bz*by + 3.d0*bc**4*dzi*dyj + 3.d0*bc**4*dzj*dyi -
     # 6*bc**3*dij*b*bz*by - 15.d0*bc**2*dij*bz*by ) / ( bc**4*b**3 )
c
c
c# di11
c
      dj11(4)=-x2m1*(2.d0*bi*bj*bc*b*bz*by+10.0d0*bi*bj*bz*by-
     # bi*bc**2*bz*dyj - bi*bc**2*by*dzj - bj*bc**2*bz*dyi -
     # bj*bc**2*by*dzi - bc**2*dij*bz*by ) / ( bc**4*b )
c
c# di20
c
      dj20(4)=x2m1*bi*bj*bz*by/(bc**4)
c
c
```

```
cd2m2
c# di00
c
c
      dj00(5)=x2m2*(bi*bj*bc**2*b**2*bx*by+10.0d0*bi*bj*bc*b*bx*by+
     # 35.d0*bi*bj*bx*by - bi*bc**3*b*bx*dyj - bi*bc**3*b*by*dxj -
     # 5.d0*bi*bc**2*bx*dyj - 5.d0*bi*bc**2*by*dxj -
     # bj*bc**3*b*bx*dyi - bj*bc**3*b*by*dxi - 5.d0*bj*bc**2*bx*dyi -
     # 5.d0*bj*bc**2*by*dxi + bc**4*dxi*dyj + bc**4*dxj*dyi -
     # 2.d0*bc**3*dij*b*bx*by - 5.d0*bc**2*dij*bx*by ) / ( bc**4*b**2 )
c
c# dim11
c
      djm11(5)=-x2m2*(3.d0*bi*bj*bc**2*b**2*bx*by+
     # 30.0d0*bi*bj*bc*b*bx*by + 105.d0*bi*bj*bx*by -
     # 3.d0*bi*bc**3*b*bx*dyj - 3.d0*bi*bc**3*b*by*dxj -
     # 15.d0*bi*bc**2*bx*dyj - 15.d0*bi*bc**2*by*dxj -
     # 3.d0*bj*bc**3*b*bx*dyi - 3.d0*bj*bc**3*b*by*dxi -
     # 15.d0*bj*bc**2*bx*dyi - 15.d0*bj*bc**2*by*dxi -
     # bc**4*dij*b**2*bx*by + 3.d0*bc**4*dxi*dyj +
     # 3.d0*bc**4*dxj*dyi - 6.d0*bc**3*dij*b*bx*by -
     # 15.d0*bc**2*dij*bx*by ) / ( bc**4*b**3 )
c
c# di11
c
      dj11(5)=-x2m2*(2.d0*bi*bj*bc*b*bx*by+10.0d0*bi*bj*bx*by-
     # bi*bc**2*bx*dyj - bi*bc**2*by*dxj - bj*bc**2*bx*dyi -
     # bj*bc**2*by*dxi - bc**2*dij*bx*by ) / ( bc**4*b )
c
c# di20
c
      dj20(5)=x2m2*bi*bj*bx*by/(bc**4)
c
c
c@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
c
 890  go to (870,872,874,876,878,880),inttyp
c
c   s-d-s
c
 870  c000=(si00(1)*sj00(1)+si00(2)*sj00(2)+si00(3)*sj00(3)+
     #       si00(4)*sj00(4) + si00(5)*sj00(5))
      cm101=(si00(1)*sjm11(1)+si00(2)*sjm11(2)+
     #     si00(3)*sjm11(3)+si00(4)*sjm11(4)+si00(5)*sjm11(5))
      cm110=(sim11(1)*sj00(1)+sim11(2)*sj00(2)+
     #     sim11(3)*sj00(3)+sim11(4)*sj00(4)+sim11(5)*sj00(5))
      cm211=(sim11(1)*sjm11(1)+sim11(2)*sjm11(2)+
     #     sim11(3)*sjm11(3)+sim11(4)*sjm11(4)+sim11(5)*sjm11(5))
      go to 882
c
c   s-d-p
c
 872  c000=(si00(1)*pj00(1)+si00(2)*pj00(2)+si00(3)*pj00(3)+
     #       si00(4)*pj00(4) + si00(5)*pj00(5))
      cm110=(sim11(1)*pj00(1)+sim11(2)*pj00(2)+
     #     sim11(3)*pj00(3)+sim11(4)*pj00(4)+sim11(5)*pj00(5))
      cm101=(si00(1)*pjm11(1)+si00(2)*pjm11(2)+si00(3)*pjm11(3)+
     #       si00(4)*pjm11(4) + si00(5)*pjm11(5))
      cm211=(sim11(1)*pjm11(1)+sim11(2)*pjm11(2)+
     #     sim11(3)*pjm11(3)+sim11(4)*pjm11(4)+sim11(5)*pjm11(5))
      c101= (si00(1)*pj11(1)+si00(2)*pj11(2)+si00(3)*pj11(3)+
     #       si00(4)*pj11(4) + si00(5)*pj11(5))
      c011= (sim11(1)*pj11(1)+sim11(2)*pj11(2)+
     #     sim11(3)*pj11(3)+sim11(4)*pj11(4)+sim11(5)*pj11(5))
```

```
c      write(60,7182)pj00(1),pjm11(1),pj11(1)
c7182  format(1x,' pj00(1)=',d15.8,' pjm11(1)=',d15.8,/,
c     #         ' pj11(1)=',d15.8)
      go to 882
c
c   p-d-p
c
 874  do 875i=1,5
      c000 =c000+pi00(i)*pj00(i)
      cm101=cm101+pi00(i)*pjm11(i)
      c101 =c101+pi00(i)*pj11(i)
      cm110=cm110+pim11(i)*pj00(i)
      cm211=cm211+pim11(i)*pjm11(i)
      c011 =c011+pim11(i)*pj11(i)
      c110 =c110+pi11(i)*pj00(i)
      c011 =c011+pi11(i)*pjm11(i)
      c211 =c211+pi11(i)*pj11(i)
 875  continue
      go to 882
c
c   s-d-d
c
 876  do 877i=1,5
      c000 =c000+si00(i)*dj00(i)
      cm110=cm110+sim11(i)*dj00(i)
      cm101=cm101+si00(i)*djm11(i)
      cm211=cm211+sim11(i)*djm11(i)
      c101 =c101+si00(i)*dj11(i)
      c011 =c011+sim11(i)*dj11(i)
      c200 =c200+si00(i)*dj20(i)
      c110 =c110+sim11(i)*dj20(i)
 877  continue
      go to 882
c
c  p-d-d
c
 878  do 879i=1,5
      c000 =c000+pi00(i)*dj00(i)
      cm101=cm101+pi00(i)*djm11(i)
      c101 =c101+pi00(i)*dj11(i)
      c200 =c200+pi00(i)*dj20(i)
      cm110=cm110+pim11(i)*dj00(i)
      cm211=cm211+pim11(i)*djm11(i)
      c011 =c011+pim11(i)*dj11(i)+pi11(i)*pjm11(i)
      c110 =c110+pim11(i)*dj20(i)+pi11(i)*dj00(i)
      c211 =c211+pi11(i)*dj11(i)
      c310 =c310+pi11(i)*dj20(i)
 879  continue
      go to 882
c
c  d-d-d
c
 880  do 881i=1,5
      c000 =c000+di00(i)*dj00(i)
      cm101=cm101+di00(i)*djm11(i)
      c101 =c101+di00(i)*dj11(i)+di20(i)*djm11(i)
      c200 =c200+di00(i)*dj20(i)+di20(i)*dj00(i)
      cm110=cm110+dim11(i)*dj00(i)
      cm211=cm211+dim11(i)*djm11(i)
      c011 =c011+dim11(i)*dj11(i)+di11(i)*djm11(i)
      c110 =c110+dim11(i)*dj20(i)+di11(i)*dj00(i)
      c211 =c211+di11(i)*dj11(i)
      c310 =c310+di11(i)*dj20(i)
      c301 =c301+di20(i)*dj11(i)
```

```
      c400 =c400+di20(i)*dj20(i)
 881  continue
c
c
 882  call vbcad(xcab)
c
cc
c  tests to determine magnitude of ddmx
c
        if(xcab.eq.0.0d0)goto1950
        if(dabs((xcab-testcab)/xcab).gt.1.0e-12.and.
     1 buftest.lt.ddtest)ddtest=buftest
c       write(60,66642)xcab
c66642 format(1x,' xcab after <d|d> ',d15.9)
 1950 return
    4 write(60,1000)jtyp
 1000 format(1h1,' *** jtyp =',i3)
 7777 write(60,1111)
 1111 format(1x,' out of range see vbca d operators')
      stop
      end
      subroutine vbcas(xcab)
c  version #2 jan 10 1985  c.woodward
c       this subroutine evaluates the terms <i!l><l!j>, l=0,
c       where i, j and the projection operator l are on different
c       centers.  the angular integrals are organized by powers of
c       r and products of spherical bessel functions, the appropriate
c       constants to each term being provided by vbca. the radial
c       integrals are then evaluated in the double precision functions
c       joo, jol, jlo, joo.
      implicit double precision(a-h,j,o-z)
      common/cnl112/cm211,cm110,cm101,c000,c011,c101,c110,c200,c211,
     #  c301,c310,c400,c411,xij,aht,bht,d,inttyp,ldstr,ldstp
      parameter(npst=20,npmx=200)
      common/int2/nlp(npmx),clp(npmx),zlp(npmx)
      common/jprm/srzi,ah,bh,dl
      dl=d
      do 100k=ldstr,ldstp
      zet3=zlp(k)+xij
      n=nlp(k)-2
      srz=dsqrt(zet3)
      srzi=1.0d0/srz
      ah= aht*srzi
      bh= bht*srzi
      ap=2.0d0*aht*srzi
      bp=2.0d0*bht*srzi
      z=ap*bp/2.0d0
      ab2= ah*ah+bh*bh
c
      ck=clp(k)
      go to (11,12,13,14,15,16),inttyp
   11 xdum=joo(2+n)*ck
      xcab=xcab+xdum
      go to 100
   12 xcab=xcab+(joo(2+n)*c000+jol(n+3)*c101)*ck
      go to 100
   13 xcab=xcab+ck*(joo(n+2)*c000+(jol(n+3)*c101+jlo(n+3)*
     #      c110+jll(n+4)*c211))
      go to 100
   14 xcab=xcab+ck*(joo(n+2)*c000+jol(n+3)*c101+joo(n+4)*c200)
      go to 100
   15 xcab=xcab+ck*(joo(n+2)*c000+jol(n+3)*c101+jlo(n+3)*c110+
     #  joo(n+4)*c200+jll(n+4)*c211+jlo(n+5)*c310)
      go to 100
```

```
   16 xcab=xcab+ck*(joo(n+2)*c000+jol(n+3)*c101+jlo(n+3)*c110+
     #   joo(n+4)*c200+j11(n+4)*c211+jlo(n+5)*c310+c301*jol(n+5)
     #   +joo(n+6)*c400)
  100 continue
      return
      end
      subroutine vbcap(xcab)
c  version #2 jan 10 1985  c.woodward
c      this subroutine evaluates the terms <i!l><l!j>, l=1,
c      where i, j and the projection operator l are on different
c      centers.  the angular integrals are organized by powers of
c      r and products of spherical bessel functions, the appropriate
c      constants to each term being provided by vbca. the radial
c      integrals are then evaluated in the double precision functions
c      joo, jol, jlo, joo.
      implicit double precision(a-h,j,o-z)
      common/cnl112/cm211,cm110,cm101,c000,c011,c101,c110,c200,c211,
     #  c301,c310,c400,c411,xij,aht,bht,d,inttyp,ldstr,ldstp
      parameter(npst=20,npmx=200)
      common/int2/nlp(npmx),clp(npmx),zlp(npmx)
      common/jprm/srzi,ah,bh,d1
      d1=d
      do 100k=ldstr,ldstp
      zet3=zlp(k)+xij
      n=nlp(k)-2
      srz=dsqrt(zet3)
      srzi=1.0d0/srz
      ah=aht*srzi
      bh=bht*srzi
      z=2.0d0*ah*bh
      ab2=ah*ah+bh*bh
c
      ck=clp(k)*3.0d0
      go to (11,12,13,14,15,16),inttyp
   11 xcab=xcab+ck*j11(n+2)*c011
      go to 100
   12 xcab=xcab+ck*(c011*j11(n+2)+c110*jlo(n+3))
      go to 100
   13 xcab=xcab+ck*(c011*j11(n+2)+c110*jlo(n+3)+c101*jol(n+3)+
     #  c200*joo(n+4))
      go to 100
   14 xcab=xcab+ck*(c011*j11(n+2)+c110*jlo(n+3)+c211*j11(n+4))
      go to 100
   15 xcab=xcab+ck*(c011*j11(n+2)+c110*jlo(n+3)+c101*jol(n+3)+
     #   joo(n+4)*c200+j11(n+4)*c211+jol(n+5)*c301)
      go to 100
   16 xcab=xcab+ck*(c011*j11(n+2)+c110*jlo(n+3)+c101*jol(n+3)+
     # joo(n+4)*c200+j11(n+4)*c211+jol(n+5)*c301+jlo(n+5)*c310
     # +c411*j11(n+6))
c
  100 continue
      return
      end
      subroutine vbcad(xcab)
c  version #2 jan 10 1985  c.woodward
c      this subroutine evaluates the terms <i!l><l!j>, l=2,
c      where i, j and the projection operator l are on different
c      centers.  the angular integrals are organized by powers of
c      r and products of spherical bessel functions, the appropriate
c      constants to each term being provided by vbca. the radial
c      integrals are then evaluated in the double precision functions
c      joo, jol, jlo, joo.
      implicit double precision(a-h,j,o-z)
      common/cnl112/cm211,cm110,cm101,c000,c011,c101,c110,c200,c211,
```

```
     # c301,c310,c400,c411,xij,aht,bht,d,inttyp,ldstr,ldstp
      parameter(npst=20,npmx=200)
      common/int2/nlp(npmx),clp(npmx),zlp(npmx)
      common/jprm/srzi,ah,bh,d1
      d1=d
      do 100k=ldstr,ldstp
      zet3=zlp(k)+xij
      n=nlp(k)
      n1=n+1
      srz=dsqrt(zet3)
      srzi=1.0d0/srz
      ah=aht*srzi
      bh=bht*srzi
      z=2.0d0*ah*bh
      ab2=ah*ah+bh*bh
c
      ck=clp(k)
      go to (11,12,13,14,15,16),inttyp
 11   xcab=xcab+ck*(c000*joo(n)+cm101*jo1(n-1)+cm110*j1o(n-1)
     #       +cm211*j11(n-2))
      go to 100
  12 xcab =xcab+ck*(c000*joo(n)+cm101*jo1(n-1)+cm110*j1o(n-1)
     #       +cm211*j11(n-2)+c101*jo1(n+1)+c011*j11(n))
      go to 100
  13 xcab=xcab+ck*(c000*joo(n)+cm101*jo1(n-1)+c101*jo1(n+1)
     #       +cm110*j1o(n-1)+cm211*j11(n-2)+c011*j11(n)+c110*
     #       j1o(n+1)+c211*j11(n+2))
      go to 100
  14 xcab =xcab+ck*(c000*joo(n)+cm101*jo1(n-1)+cm110*j1o(n-1)
     #       +cm211*j11(n-2)+c101*jo1(n+1)+c011*j11(n)+c200*
     #       joo(n+2)+c110*j1o(n+1))
      go to 100
  15 xcab =xcab+ck*(c000*joo(n)+cm101*jo1(n-1)+c101*jo1(n+1)+
     #       c200*joo(n+2)+cm110*j1o(n-1)+cm211*j11(n-2)+c011*
     #       j11(n)+c110*j1o(n+1)+c211*j11(n+2)+c310*j1o(n+3))
      go to 100
  16 xcab =xcab+ck*(c000*joo(n)+cm101*jo1(n-1)+c101*jo1(n+1)+
     #       c200*joo(n+2)+cm110*j1o(n-1)+cm211*j11(n-2)+c011*
     #       j11(n)+c110*j1o(n+1)+c211*j11(n+2)+c310*j1o(n+3)+
     #       c301*jo1(n+3)+c400*joo(n+4))
 100  continue
      return
      end
      double precision function joo(n)
c  version #2 jan 10 1985  c.woodward
c  this function evaluates integrals stemming from a three center
c  integrand.  the integrand includes two modified spherical bessel
c  functions of the first kind (1=0) with different arguements, a
c  gaussian and r**n.
      implicit double precision(a-h,j,o-z)
      integern
      common/jprm/srzi,ah,bh,d
      data srpi8/0.22155673136325d0/
      iswap=0
      n1= n +1
c
c   go to special functions if n < 11 + 12 + 2
c
      if (n .lt.2)goto500
c
      ap=2.0d0*ah
      bp=2.0d0*bh
      ab=ap*bp
      aps= ap*ap
```

206

```
      bps= bp*bp
      ab2= ah*ah+bh*bh
      abh=ap*bp/2.0d0
      go to (10,30,50,70),(n/2)
      go to 999
c
c n=2
c
  10  if(abh.gt.18.0d0)goto15
      ck=srpi8*exp(ab2+d)*(srzi**n1)
      joo=ck*4.0d0*dsinh(abh)/ab
      return
  15  ck=srpi8*(srzi**n1)
      joo=ck*2.0d0*exp(ab2+d+abh)/ab
      return
c
c n=4
c
  30  if(abh.gt.18.0d0)goto35
      ck=srpi8*exp(ab2+d)*(srzi**n1)
      if(abh.gt.0.3d0)goto31
      joo=ck*dcosh(abh)*(aps+bps+6.0d0+
     #     (aps+bps+2.0d0)*tanh3(abh))/2.0d0
      return
  31  joo=ck*(dcosh(abh)*2.0d0*ab+(aps+bps+2.0d0)*
     #     dsinh(abh))/ab
      return
  35  ck=srpi8*srzi**n1
      joo=ck*exp(ab2+d+abh)*((aps+bps)/2.0d0+ab+1.0d0)/ab
      return
c
c n=6
c
  50  if(abh.gt.18.0d0)goto55
      ck=srpi8*exp(ab2+d)*(srzi**n1)
      if(abh.lt.0.3d0)goto51
      joo=ck*(dcosh(abh)*2.0d0*ah*
     #     (3.0d0*bh+2.0d0*bh*(ah*ah+bh*bh)) +
     #     dsinh(abh)*(ah*ah*(6.0d0*bh*bh+ah*ah+3.0d0) +
     #     bh*bh*(3.0d0+bh*bh)+0.75d0))/(ah*bh)
      return
  51  joo=ck*dcosh(abh)*(ah*bh*
     #     (2.0d0*ah*ah*(5.0d0+6.0d0*bh*bh+ah*ah)+7.5d0+
     #     2.0d0*bh*bh*(5.0d0+bh*bh))+tanh3(abh)*2.0d0*ah*bh*
     #     (ah*ah*(6.0d0*bh*bh+ah*ah+3.0d0)+bh*bh*(3.0d0+bh*bh)
     #     + 0.75d0))/(ah*bh)
      return
  55  ck=srpi8*srzi**n1
      apbs=(ah+bh)**2.0d0
      joo=ck*exp(ab2+d+abh)*(apbs*(apbs+3.0d0)+.75d0)/abh
      return
c
c n=8
c
  70  if(abh.gt.18.0d0)goto75
      ck=srpi8*exp(ab2+d)*(srzi**n1)
      if (abh.lt.0.3d0)goto71
      joo=ck*(dcosh(abh)*ap*bp*(aps*(60.0d0+10.0d0*bps+3.0d0*aps)+
     1 bps*(60.0d0+3.0d0*bps)+180.0d0)+dsinh(abh)*(aps*(180.0d0+
     1 bps*(180.0d0+15.0d0*bps)+aps*(30.0d0+15.0d0*bps+aps))+
     1 bps*(180.0d0+bps*(30.0d0+bps))+120)/2.0d0)/(8.0d0*ab)
      return
  71  joo=ck*dcosh(abh)*(
     1 aps*(420.d0+bps*(220.d0+15.0d0*bps)+
```

```
     1 aps*(42.d0+15.0d0*bps+aps))+
     1 bps*(420.d0+bps*(42.0d0+bps))+840.0d0+
     1 tanh3(abh)*(aps*(180.d0+bps*(180.0d0+15.0d0*bps)+
     1 aps*(30.d0+15.0d0*bps+aps))+
     1 bps*(180.0d0+bps*(30.0d0+bps))+120.d0))/32.0d0
      return
   75 ck=srpi8*srzi**n1
      joo=ck*exp(ab2+d+abh)*(
     1 ap*(bp*(360.0d0+bps*(120.0d0+6.0d0*bps))+
     1 ap*(180.0d0+bps*(180.0d0+15.0d0*bps)+
     1 ap*(bp*(120.0d0+20.0d0*bps)+
     1 ap*(30.0d0+15.0d0*bps+ap*(6.0d0*bp+ap)))))+
     1 bps*(180.0d0+bps*(30.0d0+bps))+120.0d0)/(32.0d0*ap*bp)
      return
c
c   integrate using expansion of small arguement if neccessary
c
  500 if (bh.lt.0.1d0)goto610
      if (ah.lt.0.1d0)goto600
c
      a=2.0d0*ah
      b=2.0d0*bh
      zz=2.0d0*ah*bh
      ab2= ah*ah+bh*bh
      xp=ah+bh
      xm=bh-ah
      call dawv(dawsp,daws3p,xp)
      if(zz .gt.18.0d0)goto85
      exzp=exp(zz)
      exzm=1.0d0/exzp
      d0p=dawsp*exzp
      call dawv(dawsm,daws3m,xm)
      d0m=dawsm*exzm
      d0s=d0p+d0m
      d0d=d0p-d0m
      d1d=0.5d0*(exzp-exzm)
      if(zz.lt.0.1d0)d1d=dsinh(zz)
      joo=srpi8*exp(ab2+d)*(srzi**n1)*(b*d0d+a*d0s-2.0d0*d1d)/(ah*bh)
      return
   85 joo=srpi8*exp(ab2+d+zz)*srzi**n1*((b+a)*dawsp-1.0d0)/
     1(ah*bh)
      return
c
  600 ch=bh
      bh=ah
      ah=ch
      iswap=1
  610 z=2.0d0*bh*bh
      x2=ah*ah
      expa=exp(ah*ah+d)
      call dawv(daws,daws3,ah)
      if(ah.lt.0.3d0)goto110
      vm0=2.0d0*daws/ah
      go to 111
  110 vm0=2.0d0*(1.0d0+daws3)
  111 continue
      vm2=1.0d0
      vm4=x2+1.5d0
      vm6=x2*v5+2.5d0*vm4
      vm8=x2*v7+3.5d0*vm6
      z3=z*0.33333333333333d0
      z10=0.1d0*z
      z21=z/21.0d0
      z36=z/36.0d0
```

```
      c00=expa*(srzi**n1)*srpi8*2.0d0
      joo=c00*(vm0+z3*(vm2+z10*(vm4+z21*(vm6+z36*vm8))))
      if(iswap.eq.1)goto222
      return
  222 ch=bh
      bh=ah
      ah=ch
      return
  999 write(60,9999)
 9999 format(1x,'illegal order of r see joo')
      stop
      end
      double precision function j1o(n)
c  version #2 jan 10 1985  c.woodward
c  this function evaluates integrals stemming from a three center
c  integrand.  the integrand includes two modified spherical bessel
c  functions of the first kind (l=1 and l=0) with different arguements,
c  a gaussian and r**n.
      implicit double precision(a-h,j,o-z)
      integern
      common/jprm/srzi,ah,bh,d
      data srpi8/0.22155673136325d0/
      n1= n +1
c
c   go to special cases if n < l1 + l2 + 2
c
      if (n .lt.3)goto500
c
      ap=2.0d0*ah
      bp=2.0d0*bh
      aps=ap*ap
      bps=bp*bp
      ab=ah*bh
      ahs= ah*ah
      bhs= bh*bh
      ab2= ah*ah+bh*bh
      abt=ah*bh*2.0d0
      ap3=aps**3.0d0
      bp3=aps*aps*bps
      bpah= (ah+bh)
      bmah= (bh-ah)
c
c   otherwise evaluate j1o
c
      go to (10,999,30,999,40),(n-2)
      goto 999
c
c
  10  if(abt.gt.18.d0)goto11
      ck=srpi8*exp(ab2+d)*(srzi**n1)
      if(abt.gt.0.3d0)goto15
      j1o=2.0d0*ck*dcosh(abt)*(ahs+
     #     (ahs - 0.5d0)*tanh3(abt))/ah
      return
  15  j1o= ck*(dcosh(abt)*
     #      bh +(ah-(0.5d0/ah))*dsinh(abt))/ab
      return
  11  j1o=srpi8*exp(ab2+d+abt)*srzi**n1*(ah+bh-.5d0/ah)/abt
      return
c
c n=5
c
  30  if(abt.gt.18.0d0)goto31
      ck=srpi8*exp(ab2+d)*(srzi**n1)
```

```
      if(abt.gt.0.3d0)goto35
      jlo= ck*dcosh(abt)*
     #    (bh*(2.0d0*ahs*(1.0d0+ahs+3.0d0*bhs)-bhs-0.5d0)
     #    *tanh3(abt)+ahs*bh*(2.0d0*ahs+5.0d0+6.0d0*bhs))/ab
      return
   35 jlo=ck*dcosh(abt)*(bh*(
     #    bhs+3.0d0*ahs+0.5d0)+(ah*(ahs+1.0d0-0.25d0/ahs+
     #    bhs*(3.0d0-0.5d0/ahs)))*dtanh(abt))/ab
      return
   31 jlo=srpi8*exp(ab2+d+abt)*srzi**n1*(bpah*(1.5d0+bpah*(bpah
     1-.5d0/ah))-.25/ah)/abt
      return
c
c n=7
c
   40 if (abt.gt.18.d0)goto41
      ck=srpi8*exp(ab2+d)*(srzi**n1)
      if (abt.lt.0.3d0)goto45
      jlo=ck*(dcosh(abt)*bp*(12.0d0+bps*(12.0d0+bps)+
     1aps*(52.0d0+10.d0*bps+5.0d0*aps))+dsinh(abt)*(
     1aps*(36.0d0+bps*(48.0d0+5.0d0*bps)+
     1aps*(18.0d0+10.d0*bps+aps))-bps*(24.0d0+2.0d0*bps)-24.0d0)/ap)
     1/(16.0d0*abt)
      return
   45 jlo=ck*dcosh(abt)*(aps*(140.d0+bps*(68.0d0+5.0d0*bps)+
     1aps*(28.d0+10.0d0*bps+aps))+tanh3(abt)*(aps*(36.0d0+bps*
     1(48.d0+5.0d0*bps)+aps*(18.0d0+10.0d0*bps+aps))-bps*(24.0d0
     1+2.0d0*bps)-24.0d0))/(ap*16.0d0)
      return
   41 jlo=srpi8*exp(ab2+d+abt)*srzi**n1*(
     1ap*(bp*(12.0d0+bps*(12.0d0+bps))+
     1ap*(36.0d0+bps*(48.0d0+5.0d0*bps)+
     1ap*(bp*(52.0d0+10.0d0*bps)+
     1ap*(18.0d0+10.d0*bps+
     1ap*(5.0d0*bp+ap)))))-
     1bps*(24.0d0+2.0d0*bps)-24.0d0)/(16.0d0*aps*bp)
      return
c
c go to the expansions
c
  500 if(bh .lt.0.1d0)goto610
      if(ah .lt.0.1d0)goto600
c
      zz=2.0d0*ah*bh
      a=2.0d0*ah
      b=2.0d0*bh
      xp=ah+bh
      xm=bh-ah
      ahs= ah*ah
      bhs= bh*bh
      abt=2.0d0*ah*bh
      ab2= ah*ah+bh*bh
      call dawv(dawsp,daws3p,xp)
      call dawv(dawsm,daws3m,xm)
c
c    evaluate n=1 or n=-1
c
c      if (n .eq. -1)go to 60
c*****************************************
c*  (n=-1 is not used at this time)      *
c*****************************************
c
      if(zz .gt.18.0)goto42
      ck=srpi8*exp(d+ab2)*srzi**2.0d0
```

```
      buff=exp(zz)*4.0d0*(1.0d0-b*dawsp)
      buff=buff-exp(-zz)*4.0d0*(1.0d0-b*dawsm)
      jlo= ck*buff/(b*a*a)
      return
 42   jlo=srpi8*(srzi**2.0d0)*exp(d+ab2+zz)*4.0d0*(1.0d0-b*dawsp)
     1/(b*a*a)
      return
c
c   evaluate n=-1
c
c  60  if (zz .gt. 18.0d0)goto 61
c      d1t=-0.6666666666666d0*(ah*bh*dcosh(zz)+
c     #     (-2.0d0*ahs+bhs-1.0d0)*dsinh(zz))
c      if (zz .lt. 0.3d0) d1t=-0.6666666666666d0*
c     #   dcosh(zz)*(zz*(2.0d0*ahs+bhs-0.5d0)+zz*
c     #   (2.0d0*ahs+bhs-1.0d0)*tanh3(zz))
c      exzp=exp(zz)
c      exzm=1.0d0/exzp
c      d0p=dawsp*exzp
c      d0m=dawsm*exzm
c      d0s=d0p+d0m
c      d0d=d0p-d0m
c      d2t=d0s*4.0d0*ahs*ah/3.0d0
c      d3t=d0d*bh*(1.0d0+2.0d0*ahs-0.6666666666666d0*bhs)
c      jlo=srpi8*exp(ab2+d)*(d1t+d2t+d3t)/(bh*ahs)
cc      return
c  61  d1t=-0.3333333333333d0*(ah*bh
c     #     -2.0d0*ahs+bhs-1.0d0)
c      d2t=dawsp*(4.0d0*ahs*ah/3.0d0+bh*(1.0d0+2.0d0*ahs-
c     # 0.6666666666666d0*bhs))
c      jlo=srpi8*exp(ab2+d+zz)*(d1t+d2t)/(bh*ahs)
c      return
c
c   now do the expansions for small bh
c
 610  z=2.0d0*bh*bh
      ahs=ah*ah
      bhs=bh*bh
      expa=exp(ah*ah+d)
      call dawv(daws,daws3,ah)
      vm0=2.0d0*daws/ah
      if(ah.lt.0.3d0)goto110
      d0=(daws*(2.0d0*ah+1.0d0/ah)-1.0d0)/ahs
      d1=(1.0d0-0.5d0*vm0)/ahs
      go to 111
 110  d1=-daws3/ahs
      d0=daws3*(1.0d0/ahs+2.0d0)+2.0d0
 111  continue
      d2= 1.0d0
      d3= (2.5d0+ahs)
      d4= (8.75d0+ahs*(7.0d0+ahs))
      d5= (39.375d0+ahs*(47.25d0+ahs*(13.5d0+ahs)))
      c1= 2.0d0*bhs/3.0d0
      c2= bhs/5.0d0
      c3= 2.0d0*bhs/21.0d0
      c4= bhs/18.0d0
      c5= 2.0d0*bhs/55.0d0
      if(n .eq.-1)goto630
      jlo=2.0d0*srpi8*expa*(srzi**n1)*ah*(d1+c1*(d2+c2*(d3+c3*
     #    (d4 + c4*d5))))
      return
 630  jlo=2.0d0*srpi8*expa*ah*(d0+c1*(d1+c2*(d2+c3*
     #    (d3 + c4*(d4 + c5*d5)))))
      return
```

```
c
c   now do the expansions for small ah
c
  600 z=2.0d0*bh*bh
      ahs=ah*ah
      bhs=bh*bh
      expb=exp(bh*bh+d)
      call dawv(daws,daws3,bh)
      d0= 2.0d0*daws/bh
      if(bh.lt.0.3d0)d0=(1.0d0+daws3)*2.0d0
      d1= 1.0d0
      d2= bhs+1.5d0
      d3= 3.75d0+bhs*(5.0d0+bhs)
      d4= 13.125d0+bhs*(26.25d0+bhs*(10.5d0+bhs))
      d5= 59.0625d0+bhs*(157.5d0+bhs*(94.5d0+bhs*
     #     (18.0d0 + bhs)))
      c1= ahs*0.40d0
      c2= ahs/7.0d0
      c3= 2.0d0*ahs/27.0d0
      c4= ahs/22.0d0
      c5= 2.0d0*ahs/65.0d0
      if(n .eq.-1)goto640
      j1o= 4.0d0*ah*(srzi**n1)*srpi8*expb*(d1+c1*(d2+c2*
     #       (d3 + c3*(d4 + c4*(d5  +c5)))))/3.0d0
      return
  640 j1o= 4.0d0*ah*srpi8*expb*(d0+c1*(d1+c2*(d2+c3*
     #       (d3 + c4*(d4 + c5*d5)))))/3.0d0
      return
  999 write(60,9999)
 9999 format(1x,'odd power of r is not valid.  see j1o.')
      stop
      end
      double precision function jo1(n)
c  version #2 jan 10 1985   c.woodward
c  this function evaluates integrals stemming from a three center
c  integrand.  the integrand includes two modified spherical bessel
c  functions of the first kind (l=0 and l=1) with different arguements,
c  a gaussian and r**n.
      implicit double precision(a-h,j,o-z)
      common/jprm/srzi,ah,bh,d
      integern
      data srpi8/0.22155673136325d0/
c
      ch=ah
      ah=bh
      bh=ch
      jo1=j1o(n)
      ch=ah
      ah=bh
      bh=ch
      return
      end
      double precision function j11(n)
c  version #2 jan 10 1985   c.woodward
c  this function evaluates integrals stemming from a three center
c  integrand.  the integrand includes two modified spherical bessel
c  functions of the first kind (l=1) with different arguements,
c  a gaussian and r**n.
      implicit double precision(a-h,j,o-z)
      integern
      common/jprm/srzi,ah,bh,d
      data srpi8/0.22155673136325d0/
      iswap=0
      n1= n +1
```

```
c
c   go to special cases if n < 11 + 12 + 2
c
      if (n .lt.4)goto500
c
      ap=2.0d0*ah
      bp=2.0d0*bh
      ab=ah*bh
      ahs= ah*ah
      bhs= bh*bh
      ab2= ah*ah+bh*bh
      abt=ah*bh*2.0d0
      bpah= (ah+bh)
      bmah= (bh-ah)
c
c   otherwise evaluate j11
c
      goto(10,20,30)(n-2)/2
      go to 999
c
c   evaluate j11(4)
c
   10 if(abt.gt.18.0d0)goto11
      ck=srpi8*exp(ab2+d)*(srzi**n1)
      if(abt.lt.0.3d0)goto15
      j11= ck*(dcosh(abt)*(ahs+bhs-
     #     0.5d0) + (abt-(ahs+bhs-0.5d0)/abt)*dsinh(abt))/ab
      return
   15 j11= ck*dcosh(abt)*(abt*abt+
     #     (abt*abt-ahs-bhs+0.5d0)*tanh3(abt))/ab
      return
   11 ck=srpi8*srzi**n1
      j11= ck*exp(ab2+d+abt)*(ahs+bhs-0.5d0+
     #     (abt-(ahs+bhs-0.5d0)/abt))/abt
      return
c
c   evaluate j11(6)
c
   20 x2h=ab2-0.5d0
      x2p=ab2+0.5d0
      if(abt.gt.18.0d0)goto17
      ck=srpi8*exp(ab2+d)*(srzi**n1)
      if(abt.lt.0.3d0)goto75
      j11=ck*((x2p*(2.0d0*abt*abt-x2h)+abt*abt-ab2)*dsinh(abt)/abt+
     # (x2p*x2p+abt*abt-0.5d0)*dcosh(abt))/(abt*0.5d0)
      return
   75 j11=ck*dcosh(abt)*2.0d0*((ab2+1.5d0)*2.0d0*abt+tanh3(abt)*
     # (x2p*(2.0d0*abt*abt-x2h)+abt*abt-ab2)/abt)
      return
   17 ck=srpi8*(srzi**n1)*exp(ab2+d+abt)
      j11=ck*((x2p*(2.0d0*abt*abt-x2h)+abt*abt-ab2)/abt+
     # (x2p*x2p+abt*abt-0.5d0))/(abt)
      return
c
c  n=8
c
   30 as=ap*ap
      bs=bp*bp
      if(abt.gt.18.0d0)goto31
      ck=srpi8*srzi**n1*exp(ab2+d)
      if (abt.lt.0.30d0)goto35
      t1=(as*(bs*(140.0d0+15.0d0*bs)+36.0d0+
     1    as*(18.0d0+15.0d0*bs+as))+
     1    bs*(36.0d0+bs*(18.d0+bs))-24.d0)/64.d0
```

```
      t2=(as*(-36.d0+bs*(72.d0+bs*(45.d0+3.d0*bs))+
     1    as*(-18.d0+bs*(45.d0+10.d0*bs)+
     1    as*(3.d0*bs-1.0d0)))
     1    -bs*(36.d0+bs*(18.d0+bs))+24.d0)/(32.0d0*ap*bp)
      j11=ck*(dcosh(abt)*t1+dsinh(abt)*t2)/(ah*bh)
      return
   35 j11=ck*dcosh(abt)*(as*bs*(as*(60.0d0+10.0d0*bs+3.0d0*as)+
     1bs*(60.0d0+3.0d0*bs)+212.d0)+tanh3(abt)*(
     1as*(-36.0d0+bs*(72.0d0+bs*(45.0d0+3.0d0*bs))+
     1as*(-18.0d0+bs*(45.0d0+10.0d0*bs)+
     1as*(-1.0d0+3.0d0*bs)))-bs*(36.0d0+bs*(18.0d0+bs))+24.0d0))
     1/(16.0d0*ap*bp)
      return
   31 ck=srpi8*srzi**n1
      t1=(ap*(bp*(-24.0d0+bs*(36.0d0+bs*(18.0d0+bs)))+
     1    ap*(-72.0d0+bs*(144.0d0+bs*(90.0d0+6.0d0*bs))+
     1    ap*(bp*(36.0d0+bs*(140.0d0+15.0d0*bs))+
     1    ap*(-36.0d0+bs*(90.0d0+20.0d0*bs)+
     1    ap*(bp*(18.0d0+15.0d0*bs)+
     1    ap*(-2.0d0+6.0d0*bs+ap*bp))))))-
     1    bs*(72.0d0+bs*(36.0d0+2.0d0*bs))+48.0d0)/(64.0d0*ap*bp)
      j11=ck*exp(ab2+abt+d)*t1/abt
      return
c
c
c   check to see if ah or bh is less than 0.1
c
  500 if(bh.lt.0.1d0)goto101
      if(ah.lt.0.1d0)goto201
c
      ab2=ah*ah+bh*bh
      zz=2.0d0*ah*bh
      a=2.0d0*ah
      b=2.0d0*bh
      xp=ah+bh
      xm=bh-ah
      call dawv(dawsp,daws3p,xp)
c      d0p=dawsp*exzp
      call dawv(dawsm,daws3m,xm)
c
      go to (55,999,555)n+1
      goto 999
c
c   n=-2
c
c      ck=srpi8*exp(ab2+d)*(srzi**n1)
c  5   if (zz .lt. 0.3d0) go to 51
c      f1= 2.0d0*(dcosh(zz)*a*b*(4.0d0-b*b-a*a)+dsinh(zz)*
c     #  (a*a*(a*a-4.0d0*(b*b+2.0d0))+b*b*(b*b-8.0d0)-8.0d0))
c      go to 52
c  51  f1= 2.0d0*dcosh(zz)*(a*b*(-4.0d0+b*b*(b*b-9.0d0)
c     #    +a*a*(a*a-4.0d0*b*b-9.0d0))+(a*a*(a*a-4.0d0*(b*b+2.0d0))
c     #    +b*b*(b*b-8.0d0)-8.0d0)*tanh3(zz))
c  52  j11= 2.0d0*ck*(f1+d0p*a*a*a*(5.0d0*(b*b+2.0d0)
c     #      -a*a)+d0m*b*b*b*(5.0d0*(a*a+2.0d0)-b*b))/(15.0d0*a*a*b*b)
c      return
c
c  n=0
c
   55 if (zz.gt.18.0)goto56
      ck=srpi8*(srzi**n1)*exp(ab2+d)
      buff=exp(zz)*(a*a+b*b-a*b+2.0d0-(a**3+b**3)*dawsp)
      buff=buff-exp(-zz)*(a*a+b*b+a*b+2.0d0+(a**3-b**3)*dawsm)
      j11=ck*buff*4.0d0/(3.0d0*a*a*b*b)
```

```
      return
   56 ck=srpi8*(srzi**n1)*exp(ab2+d+zz)
      buff=(a*a+b*b-a*b+2.0d0-(a**3+b**3)*dawsp)
      j11=ck*buff*4.0d0/(3.0d0*a*a*b*b)
      return
c
c  n=2
c
  555 if(zz .gt.18.0d0)goto556
      ck=srpi8*(srzi**n1)*exp(ab2+d)
      buff=exp(zz)*(a*b-2.0d0)
      buff=buff+exp(-zz)*(a*b+2.0d0)
      j11=ck*buff*2.0d0/(a*a*b*b)
      return
  556 buff=exp(ab2+d+zz)*(a*b-2.0d0)
      ck=srpi8*(srzi**n1)
      j11=ck*buff*2.0d0/(a*a*b*b)
      return
c
c   evaluate small ah,bh
c
  201 ch=bh
      bh=ah
      ah=ch
      iswap=1
  101 z=2.0d0*bh*bh
      x2=ah*ah
      expb=exp(ah*ah+d)
      call dawv(daws,daws3,ah)
      vm0=2.0d0*daws/ah
      if(ah.lt.0.3d0)goto110
      v0= (daws*(2.0d0*ah+1.0d0/ah)-1.0d0)/x2
      v1=(1.0d0-0.5d0*vm0)/x2
      go to 111
  110 v1=-daws3/x2
      v0= (daws3*(2.0d0+1.0d0/x2)+2.0d0)
  111 continue
      vm2= 1.0d0
      v3= 1.0d0
      vm4= x2+1.5d0
      v5= x2+2.5d0
      vm6= x2*v5+2.5d0*vm4
      v7 = (x2+2.0d0)*v5+2.5d0*vm4
      v9 = (x2+3.0d0)*v7+3.5d0*vm6
      c00=expb*2.0d0
      c01=c00*bh*0.66666666666666d0
      c11=c01*ah*(srzi**n1)*srpi8
      z5= 0.2d0*z
      z14= z/14.0d0
      z27= z/27.0d0
      z44= z/44.0d0
      go to (105,106,107),(n+4)/2
c
  106 j11=c11*(v1+z5*(v3+z14*(v5+z27*(v7+z44*v9))))
       if (iswap.eq.1)goto222
       return
c
  107 continue
      vm8=x2*v7+3.5d0*vm6
      v11=(x2+4.0d0)*v9+4.5d0*vm8
c
      j11=c11*(v3+z5*(v5+z14*(v7+z27*(v9+z44*v11))))
      if(iswap.eq.1)goto222
      return
```

```
c
  105 j11= c11*(v0+z5*(v1+z14*(v3+z27*(v5+z44*v7))))
      if(iswap.eq.1)goto222
      return
c
  222 ch=bh
      bh=ah
      ah=ch
      return
c
  999 write(60,9999)
 9999 format(1x,' power of r is not valid.  see j11.')
      stop
      end
c     large uhf code for use on scalar computers
c     high speed accelerated convergence procedures
c     based on 1978 mrl code of a b kunz
c     modified by a b kunz in 1989
c     ultimately is trivially modified for large scale
c     parallel processing,with mosi changes occuring in
c     subroutine two
c     language is essentially fortran 77
c     speed up is from accelerated use of sparseness and
c     labels output on file 3
c     polyin output on file 4
c     input data on file 5
c     output on file 6
c     scratch data in arrays a1, a2,...a9, a, b, c
c     restart vectors on file 20
c     single precision mbpt output on file 30
c     ********* this code is in double precision form *********
      subroutine uhf(iblk,ilopas)
      implicit double precision (a-h,o-z)
c     *  to redimension this code change the parameters in statements  2700,2800
c     *  4800 through 5100 according to the formulae given.
c     *
c     ***********************************************************************
c     ** program limitations and dimensioning variables **
      integer*2 rdate(9)
      real*4 tyme(2)
      real*8 rdtime
      character*16 nam
      character*11 mol04(20),mol5b(20),mol16(20),mol20(20),
     1mol30(20),mol02(20)
      character*4 za,zb,zc,zd,zl,zn
      common enrep,nup,ndn/a1/a1(32400)/a2/a2(32400)/a3/a3(32400)/
     1 a4/a4(32400)/a5/a5(32400)/a6/a6(32400)/hu/hu(16290)/
     1 hd/hd(16290)/a/a(32400)/b/b(32400)/e/e(2,180)/
     1 iup/iup(180)/idn/idn(180)/rr/rr(180)/holdup/
     1 holdup(180)/holddn/holddn(180)/hold/hold(180)/sav/
     1 sav(1024)/rovr/rovr(180)
      common/files/nbas,ifile,maxbas
      common/inxcm/isnx(180)
      common/lb/ipak(1024)
      real ss
      dimension zl(20),afmt(4),za(180),zb(180),zc(180),zd(180)
      dimension zn(2)
      dimension a7(32400),a8(32400),a9(180)
      dimension work1(180),work2(180),ss(180)
      character*11 mol52(20)
      integer*2 iii,jjj
      dimension exv(1024),vxt(256,256),iii(1024),jjj(1024),
     1aij(2,256,256),sij(2,256,256)
      mol52(1)='mol5201.dat'
```

```
      mol52(2)='mol5202.dat'
      mol52(3)='mol5203.dat'
      mol52(4)='mol5204.dat'
      mol52(5)='mol5205.dat'
      mol52(6)='mol5206.dat'
      mol52(7)='mol5207.dat'
      mol52(8)='mol5208.dat'
      mol52(9)='mol5209.dat'
      mol52(10)='mol5210.dat'
      mol52(11)='mol5211.dat'
      mol52(12)='mol5212.dat'
      mol52(13)='mol5213.dat'
      mol52(14)='mol5214.dat'
      mol52(15)='mol5215.dat'
      mol52(16)='mol5216.dat'
      mol52(17)='mol5217.dat'
      mol52(18)='mol5218.dat'
      mol52(19)='mol5219.dat'
      mol52(20)='mol5220.dat'
      mol04(1)='mol0401.dat'
      mol04(2)='mol0402.dat'
      mol04(3)='mol0403.dat'
      mol04(4)='mol0404.dat'
      mol04(5)='mol0405.dat'
      mol04(6)='mol0406.dat'
      mol04(7)='mol0407.dat'
      mol04(8)='mol0408.dat'
      mol04(9)='mol0409.dat'
      mol04(10)='mol0410.dat'
      mol04(11)='mol0411.dat'
      mol04(12)='mol0412.dat'
      mol04(13)='mol0413.dat'
      mol04(14)='mol0414.dat'
      mol04(15)='mol0415.dat'
      mol04(16)='mol0416.dat'
      mol04(17)='mol0417.dat'
      mol04(18)='mol0418.dat'
      mol04(19)='mol0419.dat'
      mol04(20)='mol0420.dat'
      mol5b(1)='mol5b01.dat'
      mol5b(2)='mol5b02.dat'
      mol5b(3)='mol5b03.dat'
      mol5b(4)='mol5b04.dat'
      mol5b(5)='mol5b05.dat'
      mol5b(6)='mol5b06.dat'
      mol5b(7)='mol5b07.dat'
      mol5b(8)='mol5b08.dat'
      mol5b(9)='mol5b09.dat'
      mol5b(10)='mol5b10.dat'
      mol5b(11)='mol5b11.dat'
      mol5b(12)='mol5b12.dat'
      mol5b(13)='mol5b13.dat'
      mol5b(14)='mol5b14.dat'
      mol5b(15)='mol5b15.dat'
      mol5b(16)='mol5b16.dat'
      mol5b(17)='mol5b17.dat'
      mol5b(18)='mol5b18.dat'
      mol5b(19)='mol5b19.dat'
      mol5b(20)='mol5b20.dat'
      mol16(1)='mol1601.dat'
      mol16(2)='mol1602.dat'
      mol16(3)='mol1603.dat'
      mol16(4)='mol1604.dat'
      mol16(5)='mol1605.dat'
```

```
mol16(6)='mol1606.dat'
mol16(7)='mol1607.dat'
mol16(8)='mol1608.dat'
mol16(9)='mol1609.dat'
mol16(10)='mol1610.dat'
mol16(11)='mol1611.dat'
mol16(12)='mol1612.dat'
mol16(13)='mol1613.dat'
mol16(14)='mol1614.dat'
mol16(15)='mol1615.dat'
mol16(16)='mol1616.dat'
mol16(17)='mol1617.dat'
mol16(18)='mol1618.dat'
mol16(19)='mol1619.dat'
mol16(20)='mol1620.dat'
mol20(1)='mol2001.dat'
mol20(2)='mol2002.dat'
mol20(3)='mol2003.dat'
mol20(4)='mol2004.dat'
mol20(5)='mol2005.dat'
mol20(6)='mol2006.dat'
mol20(7)='mol2007.dat'
mol20(8)='mol2008.dat'
mol20(9)='mol2009.dat'
mol20(10)='mol2010.dat'
mol20(11)='mol2011.dat'
mol20(12)='mol2012.dat'
mol20(13)='mol2013.dat'
mol20(14)='mol2014.dat'
mol20(15)='mol2015.dat'
mol20(16)='mol2016.dat'
mol20(17)='mol2017.dat'
mol20(18)='mol2018.dat'
mol20(19)='mol2019.dat'
mol20(20)='mol2020.dat'
mol30(1)='mol3001.dat'
mol30(2)='mol3002.dat'
mol30(3)='mol3003.dat'
mol30(4)='mol3004.dat'
mol30(5)='mol3005.dat'
mol30(6)='mol3006.dat'
mol30(7)='mol3007.dat'
mol30(8)='mol3008.dat'
mol30(9)='mol3009.dat'
mol30(10)='mol3010.dat'
mol30(11)='mol3011.dat'
mol30(12)='mol3012.dat'
mol30(13)='mol3013.dat'
mol30(14)='mol3014.dat'
mol30(15)='mol3015.dat'
mol30(16)='mol3016.dat'
mol30(17)='mol3017.dat'
mol30(18)='mol3018.dat'
mol30(19)='mol3019.dat'
mol30(20)='mol3020.dat'
mol02(1)='mol0201.dat'
mol02(2)='mol0202.dat'
mol02(3)='mol0203.dat'
mol02(4)='mol0204.dat'
mol02(5)='mol0205.dat'
mol02(6)='mol0206.dat'
mol02(7)='mol0207.dat'
mol02(8)='mol0208.dat'
mol02(9)='mol0209.dat'
```

```
      mol02(10)='mol0210.dat'
      mol02(11)='mol0211.dat'
      mol02(12)='mol0212.dat'
      mol02(13)='mol0213.dat'
      mol02(14)='mol0214.dat'
      mol02(15)='mol0215.dat'
      mol02(16)='mol0216.dat'
      mol02(17)='mol0217.dat'
      mol02(18)='mol0218.dat'
      mol02(19)='mol0219.dat'
      mol02(20)='mol0220.dat'
c       ** program limitations **
      naxbas=180
      naxup=180
      naxdn=180
      mqxbas=32400
      nuptri=16290
      maxrec=360
      mplus1=16291
      maxbas=180
      maxup=180
      maxdn=180
      muptri=16290
c where:  mqxbas=maxbas*maxbas
c          (nuptri)muptri=(maxbas+1)*maxbas/2
c          maxrec=2*maxbas=file 20 double precision record size
c          (naxup)maxup=maximum number of spin up orbitals
c          (naxdn)maxdn=maximum number of spin down orbitals
c          mplus1=muptri+1
c  ** note that it is assumed throughout the code that maxup=maxdn **
      open(unit=5,file=mol04(iblk),form='formatted')
      open(unit=60,file=mol5b(iblk),form='formatted')
      open(unit=4,file=mol16(iblk),form='unformatted')
      open(unit=20,file=mol20(iblk),form='unformatted',
     1access='direct',recl=1440)
    1 format(20a4,2a4)
    2 format(/1x,20a4/8x,9a1,2x,a8)
    3 format ('1 mtu-physics code   1989 version unix in mind
     1 - double precision')
   31 format ('          version with eispack')
    4 format (20i4)
    5 format (8f10.5)
   51 format(1x,'tol=',2x, 1pe12.5/1x,'fracn=', 0pf12.5)
    6 format (80i1)
  701 format(/5x,'spin up input orbitals')
    7 format(1x,i2)
  702 format(/5x,'spin down input orbitals')
    8 format (/' up spin'/1x,80i1)
    9 format(//' initial orbital occupations')
  901 format(//' final iteration orbital occupations')
   10 format (/' down spin'/1x,80i1)
   11 format ('0 good bye for now')
   12 format ('0 converged')
   13 format ('0 not converged')
   14 format ('0 results in hartrees,1 hy=2ry=27.2ev')
   15 format ('0 state ',i4,' energy = ',f14.6,' hy ')
   16 format (1x,10f11.6)
   17 format ('0 total energy = ',f16.6,' hy')
  172 format('0 total system spin =',f10.6)
  173 format('0 ***problem in spin computation: <s**2> =',f10.6)
   18 format ('0 self consistency (energy, sqdif) = ',1pe12.3,
     1', ',e12.3)
   19 format ('0 iteration number = ',i3)
   21 format(4a8)
```

```
  22  format(16(1x,5d15.8/))
  23  format(2a4)
  24  format(1x,2a4)
  241 format(//5x,'nuclear repulsion energy =',f18.9)
  242 format(4x,'lowest overlap eigenvalue =',f18.9/)
  245 format('0 energy=',f16.7,' hy'//)
  255 format('0    nup     ndn    nbas    maxit    norm',
     1'    idat    iout    ifile    ivect'/3x,9(i4,4x)//////)
  256 format(/' %warning - iterating to an excited state solution
     1 while sorting by overlap may lead to unpredictable results')
  260 format(//' occupied orbitals - spin up')
  265 format(//' virtual orbitals - spin up')
  270 format(//' occupied orbitals - spin down')
  275 format(//' virtual orbitals - spin down')
6300  format(1x,'enter the name of the uhfabk input file (file 5)')
32400  format(a16)
61024  format(1x,'enter the name of the uhfabk information output file
     1 (file 6)')
6600  format(1x,'enter the name of the polyin output file (file 4)')
6700  format(1x,'enter the name of the uhfabk restart file (file 20)')
6800  format(1x,'enter the name of the mbpt data file (file 30)')
6900  format(1x,'enter the name of the labels output file (file 3)')
        do 101 j=1,maxbas
  101   isnx(j)=(j*(j-1))/2
c      isnx is used in computing the index of elements of an upper triangular
c      matrix stored as a linear array
        iflag=0
c      iflag=1 for first set of data
c           >1 for subsequent sets off the same data file
c      (beware: file 20 will be reused)
2000    read(5,1,end=1054)(zl(i),i=1,20),(zn(j), j=1,2)
        if(zl(1).eq.zn(1) .and. zl(2).eq.zn(2)) go to 1054
        iflag=iflag+1
        write(60,3)
        print 3
        write(60,31)
        print 31
c      date and time are decsystem-20 macros
        write(60,2)(zl(i),i=1,20),(rdate(j),j=1,9),rdtime
        print 2,(zl(i),i=1,20),(rdate(j),j=1,9),rdtime
        do 25 i=1,maxup
        iup(i)=0
  25    idn(i)=0
        iter=0
        iusen=0
        idsen=0
        eto=0.0
        etn=0.0
        read(5,4)nup,ndn,nbas,maxit,norm,idat,iout,ifile,ivect
      ifile=1
        if(ilopas.ne.0)idat=1
        write(60,255)nup,ndn,nbas,maxit,norm,idat,iout,ifile,ivect
        print 255,nup,ndn,nbas,maxit,norm,idat,iout,ifile,ivect
        imbpt=iout/2
        iout=iout-2*imbpt
        istovp=ivect/2
        isqd=ivect-2*istovp
        if (istovp.eq.1.and.norm.ne.0) write(60,256)
        if(imbpt.ne.1)go to 430
      open(unit=30,file=mol30(iblk),form='unformatted',
     1access='direct',recl=720)
430     if(iflag.gt.1)go to 44
        if(ifile.eq.0)go to 44
      open(unit=3,file=mol02(iblk),
```

```
     1form='unformatted')
44      ntot=nup+ndn
        isz=nbas*nbas
        if (ntot.eq.0)go to 1054
        read(5,5)tol,fracn
        if (tol.eq.0.0)tol=.1e-5
        if(fracn.eq.0.0)fracn=1.0
        write(60,51)tol,fracn
        print 51,tol,fracn
        if (maxit.eq.0)maxit=20
c        initialize scratch arrays
        nnn=maxbas*maxbas
        do 45 i=1,nnn
        a1(i)=0.0
        a2(i)=0.0
        a3(i)=0.0
        a4(i)=0.0
        a5(i)=0.0
        a6(i)=0.0
        a7(i)=0.0
        a8(i)=0.0
        a(i)=0.0
45      b(i)=0.0
c        input vectors loaded into a1 and a2, and output to file 6 between statements 47 a
nd 290
47      if (idat.eq.0) go to 50
        if(idat.gt.0) go to 32
c      null input vectors
        id20=1
        do 28 j=1,nup
        write(20,rec=id20)(a1(i), i=1,nbas)
 28   id20=id20+1
        if(ndn.eq.0)go to 32
        id20=181
        do 29 j=1,ndn
        write(20,rec=id20)(a2(i), i=1,nbas)
  29  id20=id20+1
        go to 32
c      read vectors off of file 5
50      write(60,701)
        do 282 i=1,nup
      za(i)='i'
      zc(i)='nup'
        read(5,5)(rr(j), j=1,nbas)
        write(60,24)za(i),zc(i)
        write(60,22)(rr(j), j=1,nbas)
        do 282 k2=1,nbas
        k3=k2+(i-1)*nbas
282     a1(k3)=rr(k2)
        if(ndn.eq.0)go to 290
        write(60,702)
        do 285 i=1,ndn
      zb(i)='i'
      zd(i)='ndn'
        read(5,5)(rr(j), j=1,nbas)
        write(60,24)zb(i),zd(i)
        write(60,22)(rr(j), j=1,nbas)
        do 285 k5=1,nbas
        n5=(i-1)*nbas+k5
285     a2(n5)=rr(k5)
32      if(idat.eq.0) go to 290
c      read input vectors off of file 20
        id20=1
        l1=1
```

```
        nbas3=nbas
        do 288 j5=1,nup
        read(20,rec=id20)(a1(j), j=l1,nbas3)
      id20=id20+1
        l1=l1+nbas
288     nbas3=nbas3+nbas
        if(ndn.eq.0)go to 290
        id20=181
        l2=1
        nbas4=nbas
        do 2882 j6=1,ndn
        read(20,rec=id20)(a2(j), j=l2,nbas4)
      id20=id20+1
        l2=l2+nbas
2882    nbas4=nbas4+nbas
290     continue
c      if norm .eq. 0, set orbital occupations for lowest nup and ndn orbitals to one, all
 others
        if(norm.ne.0)go to 38
        do 39 i=1,nbas
        iup(i)=0
        if(i.le.nup)iup(i)=1
        idn(i)=0
        if(i.le.ndn)idn(i)=1
  39    continue
        go to 40
c      read in orbital occupations if norm .eq. 1
  38    read(5,6)(iup(i),i=1,nbas)
        if(ndn.ne.0)read(5,6)(idn(i),i=1,nbas)
  40    continue
        if (nbas.gt.maxbas)stop 'too damn big'
c        output orbital occupations to file 6
        write(60,9)
        write(60,8)(iup(i),i=1,nbas)
        if(ndn.ne.0)write(60,10)(idn(i),i=1,nbas)
c      rtime is a decsystem-20 macro which returns batch time left for the job to run
c       call rtime(rta)
        call one
c      normalize the input vectors
        if(idat.ne.0)go to 202
        do 200 i=1,nup
        l1=(i-1)*nbas
        l2=l1+1
        call gdotpr(a1(l2),a3,a1(l2),0,sum,nbas)
        sum=dsqrt(sum)
        do 200 j=1,nbas
        k2=l1+j
200     a1(k2)=a1(k2)/sum
        do 201 i=1,ndn
        l1=(i-1)*nbas
        l2=l1+1
        call gdotpr(a2(l2),a3,a2(l2),0,sum,nbas)
        sum=dsqrt(sum)
        do 201 j=1,nbas
        k2=l1+j
201     a2(k2)=a2(k2)/sum
202     if(idat.ne.1)go to 203
        if(norm.ne.2)go to 203
        do 2022 i=1,nup
        if(iup(i).eq.0)iupout=i
        if(iup(i).eq.0)go to 2023
2022    continue
        iusen=1
2023    if(ndn.eq.0)go to 203
```

```
      do 2025 i=1,ndn
      if(idn(i).eq.0)idnout=i
      if(idn(i).eq.0)go to 2026
2025  continue
      idsen=1
2026  continue
      if(iusen.ne.0)go to 2050
      do 2030 j=1,nbas
2030  holdup(j)=a1((iupout-1)*nbas+j)
2050  if(idsen.ne.0)go to 203
      do 2060 j=1,nbas
2060  holddn(j)=a2((idnout-1)*nbas+j)
203   do 60 i=1,nbas
      is=(i-1)*nbas
      do 60 j=1,nbas
      k10=is+j
60    a8(k10)=a3(k10)
c     put overlap matrix into packed upper triangular form
      k=1
      do 100 j=2,nbas
      l=nbas*(j-1)
      do 100 i=1,j
      l=l+1
      k=k+1
100   a8(k)=a8(l)
c     compute eigenvalues and eigenvectors of the overlap matrix
      call rsp (nbas,nbas,mqxbas,a8,a9,1,a7,work1,
     & work2,ierr)
      if (ierr.ne.0) stop 'trouble in eigenvalue finder.'
c     reverse the order of these eigenvalues and vectors
      limit=nbas/2
      do 150 i=1,limit
      temp=a9(i)
      a9(i)=a9(nbas-i+1)
      a9(nbas-i+1)=temp
      i1=nbas*(i-1)
      in=nbas*(nbas-i)
      do 150 j=1,nbas
      temp=a7(i1+j)
      a7(i1+j)=a7(in+j)
      a7(in+j)=temp
150   continue
c     output nuclear repulsion energy and lowest overlap eigenvalue
      write(60,241) enrep
      write(60,242) a9(nbas)
c     iteration loop begins here
      if(idat.lt.0)then
c     provide trial eigenvalues and eigenfunctions
c     copy h from a4 to a5
      do 3001 i=1,nbas*nbas
      a8(i)=a7(i)
 3001 a5(i)=a4(i)
      call xroot(nbas,a5,a9,a8,rr,b,work1,work2)
      call norms(nbas)
c     trial vectors obtained
c     write to disc file 20 and toal and a2
      do 3002 i=1,nbas
      do 3002 j=1,nbas
      k=(i-1)*nbas+j
      a1(k)=b(k)
 3002 a2(k)=b(k)
      id20=1
      i20=181
      do 3003 i=1,nbas
```

```
      l1=(i-1)*nbas+1
      l2=l1-1+nbas
      write(20,rec=id20)(b(k),k=l1,l2)
 3003 write(20,rec=id20)(b(k),k=l1,l2)
c     begin iterations
      endif
 1000   iter=iter+1
c      shall we do another iteration?
        if(iter.gt.maxit)go to 1001
        if(iter.eq.1)go to 761
c       call rtime(rtb)
c       elapst=rta-rtb
c       rta=rtb
c       if(rta.lt.(1.2*elapst))go to 990
  761   continue
c     decision loop starts here
      call two
c
c**********************************************************************
c
c     start of lopas insert
c
c**********************************************************************
c
      do 899 i=1,256
      do 899 j=1,256
 899  vxt(i,j)=0.0
      if (ilopas.eq.0) go to 800
      open(unit=52,file=mol52(iblk),form='unformatted')
 801  read(52)ii,ifrst,iii,jjj,exv
      do 802 i=1,ii
      vxt(iii(i),jjj(i))=exv(i)
 802  vxt(jjj(i),iii(i))=exv(i)
      if(ifrst.eq.0)go to 801
      close(unit=52)
c     form sij and aij here
c     spin up sij first
      do 803 i=1,nbas
      do 803 j=1,nup
      sij(1,i,j)=0.0
      do 803 k=1,nbas
 803  sij(1,i,j)=sij(1,i,j)+a1((j-1)*nbas+k)*a3((i-1)*nbas+k)
c     spin down sij now
      do 804 i=1,nbas
      do 804 j=1,ndn
      sij(2,i,j)=0.0
      do 804 k=1,nbas
 804  sij(2,i,j)=sij(2,i,j)+a2((j-1)*nbas+k)*a3((i-1)*nbas+k)
c     spin up aij now
      do 805 i=1,nup
      do 805 j=1,nup
      aij(1,i,j)=0.0
      do 805 k=1,nbas
      do 805 l=1,nbas
 805  aij(1,i,j)=aij(1,i,j)+vxt(k,l)*a1((i-1)*nbas+k)*
     1a1((j-1)*nbas+l)
c     spin down aij now
      do 806 i=1,ndn
      do 806 j=1,ndn
      aij(2,i,j)=0.0
      do 806 k=1,nbas
      do 806 l=1,nbas
 806  aij(2,i,j)=aij(2,i,j)+vxt(k,l)*a2((i-1)*nbas+k)*
     1a2((j-1)*nbas+l)
```

```
c     matrix elements formed, add to hamiltonian
      do 807 i=1,nbas
      do 807 j=1,i
      sup=vxt(i,j)
      sdn=vxt(i,j)
c     form spin up part
      do 808 k=1,nup
      do 808 l=1,nup
 808  sup=sup-sij(1,i,k)*sij(1,j,l)*aij(1,k,l)
c     form spin down part
      do 809 k=1,ndn
      do 809 l=1,ndn
 809  sdn=sdn-sij(2,i,k)*sij(2,j,l)*aij(2,k,l)
c     all hamiltonian parts formed, add to hamiltonian
      m12=(i-1)*nbas+j
      m21=(j-1)*nbas+i
      if (i.eq.j)then
          a5(m12)=a5(m12)+sup
          a6(m12)=a6(m12)+sdn
      else
          a5(m12)=a5(m12)+sup
          a5(m21)=a5(m21)+sup
          a6(m12)=a6(m12)+sdn
          a6(m21)=a6(m21)+sdn
      endif
 807  continue
 800  continue
c
c**********************************************************************
c
c     end of lopas insert
c
c**********************************************************************
c

c     end of decision loop
c     small integrals will not be updated untill
c     large terms are selfconsistent
c      at this point in the program the scratch arrays contain
c      a1: spin up input vectors
c      a2: spin down input vectors
c      a3: overlap matrix
c      a4: one electron kinetic + potential energy
c      a5: the full up spin hamiltonian
c      a6: the full down spin hamiltonian
c      a7: the eigenvectors of the overlap matrix
c      a8: garbage (scratch matrix)
c      a9: the eigenvalues of the overlap matrix
c     copy overlap eigenvectors into a8
        nn=nbas*nbas
      if(norm.eq.0.and.nup.eq.ndn)then
      do 3000 i=1,nn
      a6(i)=a5(i)
 3000 continue
      endif
        do 700 k=1,nn
700     a8(k)=a7(k)
c     diagonalize spin up hamiltonian
        call xroot(nbas,a5,a9,a8,rr,b,work1,work2)
        call norms(nbas)
c      don't reverse the order of the eigenvalues and eigenvectors
c      copy eigenvectors from b to a and a5 and eigenvalues from
c      rr to ss and e
        k=0
```

```
      do 63 i=1,nbas
      ss(i)=rr(i)
      e(1,i)=rr(i)
      do 63 j=1,nbas
      k=k+1
      a5(k)=b(k)
  63  a(k)=b(k)
c     depopulate desired orbitals
      if(norm.eq.0) go to 66
      if(norm.eq.1)go to 605
      if(iusen.ne.0)go to 605
      call rover(0,nbas,nup)
605   ii=0
      i1=0
      i2=0
      do 64 i=1,nnn
64    a(i)=0.0
      do 67 i=1,nbas
      if(iup(i).eq.0)go to 675
      ii=ii+1
      e(1,ii)=rr(i)
      do 68 j=1,nbas
  68  a((ii-1)*nbas+j)=b((i-1)*nbas+j)
      go to 67
675   if(ii.lt.nup)go to 688
      e(1,i)=rr(i)
      do 676 jj=1,nbas
676   a((i-1)*nbas+jj)=b((i-1)*nbas+jj)
      go to 67
688   i2=i2+1
      sav(i2)=i
  67  continue
      if (ii.lt.nup) stop 'core orbital deleted and not replaced.'
      if(i2.eq.0)go to 66
      do 677 i3=1,i2
      i4=sav(i3)
      k9=nup+i3
      e(1,k9)=rr(i4)
      do 677 jj=1,nbas
677   a((k9-1)*nbas+jj)=b((i4-1)*nbas+jj)
66    id20=1
      call srtovp(a1,a,e,1,istovp)
c     new vectors equal weighted average of new computed
c     vectors and old vectors
      do 56 i=1,nbas
      m12=(i-1)*nbas
      k1=m12+nbas
      if (fracn.eq.1.0) goto 502
      do 55 j=1,nbas
      k1=m12+j
55    a(k1)=a(k1)*fracn+(1.-fracn)*a1(k1)
c     normalize "total" spin up vectors
      m15=m12+1
      call gdotpr(a(m15),a3,a(m15),0,sum,nbas)
      sum=dsqrt(sum)
      do 1024 j=1,nbas
      k2=m12+j
1024   a(k2)=a(k2)/sum
502   i1=m12+1
      write(20,rec=id20)(a(l1), l1=i1,k1)
  56 id20=id20+1
      call sqdif(nup,a1,a,work1,work2,squp)
      nn=nbas*nbas
      do 504 k=1,nn
```

```
504     al(k)=a(k)
        sqdn=0.
        if(ndn.eq.0) go to 90
c     copy overlap eigenvectors into a8
        do 81 k=1,nn
81      a8(k)=a7(k)
c     diagonalize spin down hamiltonian
        call xroot(nbas,a6,a9,a8,rr,b,work1,work2)
        call norms(nbas)
c      don't reverse the order of the eigenvectors and eigenvalues
c      copy eigenvectors from b to a and a6 and eigenvalues from
c      rr to e
c      note values in rr must be preserved for file 30
        k=0
        do 73 i=1,nbas
        e(2,i)=rr(i)
        do 73 j=1,nbas
        k=k+1
        a6(k)=b(k)
  73    a(k)=b(k)
c      depopulate desired orbitals
        if(norm.eq.0)go to 76
        if(norm.eq.1)go to 705
        if(idsen.ne.0)go to 705
        call rover(1,nbas,ndn)
705     ii=0
        i1=0
        i2=0
        do 74 i=1,nnn
74      a(i)=0.0
        do 77 i=1,nbas
        if(idn(i).eq.0)go to 775
        ii=ii+1
        e(2,ii)=rr(i)
        do 78 j=1,nbas
  78    a((ii-1)*nbas+j)=b((i-1)*nbas+j)
        go to 77
775     if(ii.lt.ndn)go to 778
        e(2,i)=rr(i)
        do 776 jj=1,nbas
776     a((i-1)*nbas+jj)=b((i-1)*nbas+jj)
        go to 77
778     i2=i2+1
        sav(i2)=i
  77    continue
        if (ii.lt.ndn) stop 'core orbital deleted and not replaced.'
        if(i2.eq.0)go to 76
        do 777 i3=1,i2
        i4=sav(i3)
        k9=ndn+i3
        e(2,k9)=rr(i4)
        do 777 jj=1,nbas
777     a((k9-1)*nbas+jj)=b((i4-1)*nbas+jj)
76      id20=181
        call srtovp(a2,a,e,2,istovp)
c      new vectors equal weighted average of new computed
c      vectors and old vectors
        do 87 i=1,nbas
        m12=(i-1)*nbas
        k1=m12+nbas
        if (fracn.eq.1.0) goto 503
        do 86 j=1,nbas
        k1=m12+j
86      a(k1)=fracn*a(k1)+(1.-fracn)*a2(k1)
```

```
c      normalize "total" spin down vectors
        m15=m12+1
        call gdotpr(a(m15),a3,a(m15),0,sum,nbas)
        sum=sqrt(sum)
        do 501 j=1,nbas
        k2=m12+j
501     a(k2)=a(k2)/sum
503     m14=m12+1
        write(20,rec=id20)(a(l), l=m14,k1)
   87  id20=id20+1
        call sqdif(ndn,a2,a,work1,work2,sqdn)
        nn=nbas*nbas
        do 505 k=1,nn
505     a2(k)=a(k)
c       write out data of interest for each iteration
   90   eto=etn
        call et(etn,nup,ndn)
        edif=dabs(etn-eto)
        sqdf=squp+sqdn
        write(60,19)iter
        print 19,iter
        write(60,18)edif,sqdf
        print 18,edif,sqdf
        write(60,245)etn
        print 245,etn
c       if convergence has not occurred, return to beginning of
c       iteration loop
        if (isqd.eq.1) edif=sqdf
        if(edif.gt.tol) go to 1000
c     test other factors of selfconsistncy here
c      if(iter.gt.1)then
c         iter=0
c         go to 1000
c      endif
c     end of extended convergence test
        write(60,12)
        print 12
        go to 1002
 1001   write(60,13)
 1002   write(60,14)
        write(60,17)etn
c      compute and write spin
c      use a8 as a scratch variable
        sum=0.75*dfloat(nup+ndn)+0.25*dfloat(nup*(nup-1)+ndn*(ndn-1))
     1 -0.5*dfloat(nup*ndn)
        do 620 i=1,nup
        ii=(i-1)*nbas
        do 620 l=1,nbas
        il=ii+l
        ll=(l-1)*nbas
        y=0.0
        do 610 k=1,nbas
        ik=ii+k
        lk=ll+k
610     y=y+a3(lk)*a1(ik)
620     a8(il)=y
        sum1=0.0
        do 720 i=1,nup
        ii=(i-1)*nbas
        do 720 j=1,ndn
        jj=(j-1)*nbas
        y=0.0
        do 730 l=1,nbas
        il=ii+l
```

```
         jl=jj+1
730      y=y+a2(jl)*a8(il)
720      suml=suml+y*y
         sum=sum-suml
         spin=0.
         if (sum.gt.-0.25) spin=dsqrt(0.25+sum)-0.5
         if (sum.gt.-0.25) write(60,172) spin
         if (sum.le.-0.25) write(60,173) sum
         if (norm.ne.2) goto 1006
         write(60,901)
         write(60,8) (iup(j),j=1,nbas)
         if (ndn.ne.0) write(60,10) (idn(j),j=1,nbas)
c        write out orbital compositions
1006     continue
         write(60,260)
         do 1003 i=1,nup
         write(60,15)i,e(1,i)
1003     write(60,16)(a1((i-1)*nbas+j), j=1,nbas)
         if(iout.ne.0)go to 1100
         ncom=nup+1
         write(60,265)
         do 1110 i=ncom,nbas
         write(60,15)i,e(1,i)
1110     write(60,16)(a1((i-1)*nbas+j), j=1,nbas)
1100     if(ndn.eq.0)go to 1004
         write(60,270)
         do 1005 i=1,ndn
         write(60,15)i,e(2,i)
1005     write(60,16)(a2((i-1)*nbas+j), j=1,nbas)
         if(iout.ne.0)go to 1004
         ncom=ndn+1
         write(60,275)
         do 1105 i=ncom,nbas
         write(60,15)i,e(2,i)
1105     write(60,16)(a2((i-1)*nbas+j), j=1,nbas)
1004     if (imbpt.ne.1) goto 0909
c       prepare data file for program mbpt
         id30=1
         write(30,rec=id30)ss
       id30=id30+1
         do 333 i=1,maxbas
333      ss(i)=rr(i)
         write(30,rec=id30)ss
       id30=id30+1
         do 335 i=1,nbas
         ii=(i-1)*nbas
         do 334 j=1,nbas
334      ss(j)=a5(ii+j)
         write(30,rec=id30)ss
 335   id30=id30+1
         id30=3+nbas
         do 337 i=1,nbas
         ii=(i-1)*nbas
         do 336 j=1,nbas
336      ss(j)=a6(ii+j)
         write(30,rec=id30)ss
 337   id30=id30+1
909      call mulpop
         goto 1000
1054     continue
         write(60,11)
         close(unit=5)
         close(unit=4)
         close(unit=20)
```

```
         close(unit=3)
 8765 format(1x,f18.6,4x,f18.6)
         return
         end
      subroutine mulpop
c     define needed variables
      parameter (naxbas=180)
      parameter (naxoc=300)
      implicit real*8 (a-h,o-z)
      common /a1/a1(1)/a2/a2(1)/a3/a3(1)/files/nbas,ifile,maxbas
      common enrep,nup,ndn
      real*8 pops(naxbas,2),opops(naxbas),apops(naxoc,2)
      integer nocc(2),ncntr(naxbas),kcntr(naxbas),ktype(naxbas)
      logical skip
c     calculate mullikan populations of orbitals and atoms
c     read in basis set information for mulpop
      rewind4
      read(4)
      read(4)nbasns,(ndum,ndum,ncntr(i),ndum,kcntr(i),ktype(i),
     1i=1,nbasns)
      read(4)
      read(4)noc
      rewind4
      skip=(mod(nbas,10).eq.0)
      do 29290 il=1,2
      do 1010 i=1,nbas
 1010 pops(i,il)=0.d0
      nocc(il)=nup
      if(il.eq.2) nocc(il)=ndn
      if(nocc(il).eq.0) go to 29290
      do 6060 ic=1,nocc(il)
      do 1111 i=1,nbas
 1111 opops(i)=0.d0
      ii=0
      do 5050 i=1,nbas
      tiic=a1((ic-1)*nbas+i)
      if(il.eq.2) tiic=a2((ic-1)*nbas+i)
      if(tiic.eq.0.d0) go to 5050
      temp=0.d0
81851 format(1h0,23x,'spin up orbitals only')
81852 format(1h0,23x,'spin down orbitals only')
      do 4040 j=1,nbas
      tjic=a1((ic-1)*nbas+j)
      if(il.eq.2) tjic=a2((ic-1)*nbas+j)
4040  temp=temp+tiic*tjic*a3((i-1)*nbas+j)
      opops(i)=opops(i)+temp
 5050 continue
      do 19191 i=1,nbas
19191 pops(i,il)=pops(i,il)+opops(i)
 6060 continue
c     write out mulliken populations over basis functions
c     in orbital ten column unit
      n=1
      write(60,8185)
 8185 format(1h0/15x,'mulliken population over basis functions')
      if(il.eq.2) go to 19196
      write(60,81851)
      go to 19197
19196 write(60,81852)
19197 continue
      if(nbas.le.10) go to 21215
      do 21210 m=10,nbas,10
      write(60,19190) (i,i=n,m)
      write(60,19200) (kcntr(i),i=n,m)
```

```
      write(60,19210) (ktype(i),i=n,m)
19190 format(//5x,'function',2x,10(5x,i2,3x))
19200 format(5x,'center',4x,10(5x,a4,1x))
19210 format(5x,'type',6x,10(5x,a4,1x))
      write(60,20200) (pops(i,il),i=n,m)
20200 format(5x,'population',10(2x,f8.4))
21210 n=m+1
      if(skip) go to 22220
21215 m=nbas
      write(60,19190) (i,i=n,m)
      write(60,19200) (kcntr(i),i=n,m)
      write(60,19210) (ktype(i),i=n,m)
      write(60,20200) (pops(i,il),i=n,m)
22220 continue
c     calculate mulliken populations over atoms
      do 23230 i=1,noc
23230 apops(i,il)=0.d0
      do 25250 i=1,nbas
      nc=ncntr(i)
25250 apops(nc,il)=apops(nc,il)+pops(i,il)
c     write out mulliken populations over atoms
      write(60,26260)
26260 format(///5x,'mulliken populations for atoms'//5x,'center',5x,
     *'population'/)
      do 28280 i=1,noc
      do 27270 j=1,nbas
      lcntr=kcntr(j)
      if(ncntr(j).eq.i)go to 27275
27270 continue
      go to 28280
27275 write(60,28285)lcntr,apops(i,il)
28280 continue
28285 format(6x,a4,7x,f8.4)
29290 continue
c     write out mulliken populations over basis functions
c     in orbital ten column unit
      do 29295 i=1,nbas
29295 pops(i,1)=pops(i,1)+pops(i,2)
      n=1
      write(60,8185)
      write(60,81853)
81853 format(1h0,23x,'total populations')
      if(nbas.le.10) go to 41415
      do 41410 m=10,nbas,10
      write(60,19190) (i,i=n,m)
      write(60,19200) (kcntr(i),i=n,m)
      write(60,19210) (ktype(i),i=n,m)
      write(60,20200) (pops(i,1),i=n,m)
41410 n=m+1
      if(skip) go to 44440
41415 m=nbas
      write(60,19190) (i,i=n,m)
      write(60,19200) (kcntr(i),i=n,m)
      write(60,19210) (ktype(i),i=n,m)
      write(60,20200) (pops(i,1),i=n,m)
44440 continue
c     calculate mulliken populations over atoms
      do 43430 i=1,noc
43430 apops(i,1)=apops(i,1)+apops(i,2)
c     write out mulliken populations over atoms
      write(60,26260)
      do 48480 i=1,noc
      do 47470 j=1,nbas
      lcntr=kcntr(j)
```

```
      if (ncntr(j).eq.i)go to 47475
47470 continue
      go to 48480
47475 write(60,28285)lcntr,apops(i,1)
48480 continue
      return
      end
c       this subroutine reads the one electron integrals and
c         fills the overlap matrix and the spin up and spin
c         down hamiltonians
        subroutine one
        implicit double precision(a-h,o-z)
        integer*2 iil(1024),jjl(1024),itgl(1024)
        common/files/nbas,ifile
        common enrep/al/al(1)/a2/a2(1)/a3/a3(1)/a4/a4(1)/a5/a5(1)/a6/
     1  a6(1)/a/a(1)/b/b(1)/e/e(2,1)/iup/iup(1)/idn/idn(1)/
     1  rr/rr(1)/sav/sav(1024)
        common/lb/ipak(1024)/inxcm/isnx(1)
        dimension label(18)
        rewind 4
        read(4)label
        read(4)nbfn
        if(nbfn.ne.nbas)go to 90
        read(4)
        read(4)
        read(4)enrep
        write(60,100)label
100     format(/18a4)
        read(4)nlab
20      read(4)nints,last,iil,jjl,itgl,sav
c      integrals read in batches of 1024. nints is how many there are in this batch, last
is
c      a sentry for the last batch, ipak is the packed label, sav is the integral.
c      read overlap integrals into a3
        do 30 m=1,nints
        i=iil(m)
        j=jjl(m)
        itag=itgl(m)
        a3((i-1)*nbas+j)=sav(m)
30      a3((j-1)*nbas+i)=sav(m)
        if(last.eq.0)go to 20
c      read kinetic energy integrals into a4
        read(4)nlab
35      read(4)nints,last,iil,jjl,itgl,sav
        do 40 m=1,nints
        i=iil(m)
        j=jjl(m)
        itag=itgl(m)
        a4((i-1)*nbas+j)=sav(m)
40      a4((j-1)*nbas+i)=sav(m)
        if(last.eq.0)go to 35
c      read potential energy integrals and add them into a4
        read(4)nlab
45      read(4)nints,last,iil,jjl,itgl,sav
        do 50 m=1,nints
        i=iil(m)
        j=jjl(m)
        itag=itgl(m)
        k5=(i-1)*nbas+j
        a4(k5)=a4(k5)+sav(m)
50      a4((j-1)*nbas+i)=a4(k5)
        if(last.eq.0)go to 45
        return
90      write(60,120)nbfn,nbas
```

```
120     format(115x,'nbf from tape:',i3,'   not equal to nbf input:',i3)
        stop
        end
c       this subroutine normalizes the orbital vectors
        subroutine norms(nbas)
        implicit double precision (a-h,o-z)
        common/a/a(1)/b/b(1)/a1/a1(1)/a2/a2(1)/a3/a3(1)/
     1  a4/a4(1)/a5/a5(1)/a6/a6(1)
        do 2 i=1,nbas
        m12=(i-1)*nbas
        m15=m12+1
        call gdotpr(b(m15),a3,b(m15),0,sum,nbas)
        sum=dsqrt(sum)
        do 2 j=1,nbas
        k2=m12+j
    2   b(k2)=b(k2)/sum
        return
        end
c       this subroutine identifies the core orbital
c          to be depopulated
        subroutine rover(i1,nbas,n)
        implicit double precision (a-h,o-z)
        common/iup/iup(1)/idn/idn(1)/holdup/holdup(1)/holddn/holddn(1)/
     1  hold/hold(1)/a3/a3(1)/rovr/rovr(1)/a/a(1)
        if(i1.eq.1)go to 3
        do 2 j1=1,nbas
2       hold(j1)=holdup(j1)
        go to 5
3       do 4 j1=1,nbas
4       hold(j1)=holddn(j1)
5       continue
        do 6 i=1,nbas
        in1=(i-1)*nbas+1
6       call gdotpr(hold,a3,a(in1),0,rovr(i),nbas)
        omax=0.0
        do 10 i=1,nbas
        tst=dabs(rovr(i))
        if(tst.gt.omax)ielim=i
        if(tst.gt.omax)omax=tst
10      continue
      if(i1.ne.0)go to 20
      do 15 i=1,nbas
   15 iup(i)=0
      do 16 i=1,n
   16 iup(i)=1
      iup(ielim)=0
      if(ielim.le.n)iup(n+1)=1
      go to 40
   20 do 25 i=1,nbas
   25 idn(i)=0
      do 26 i=1,n
   26 idn(i)=1
      idn(ielim)=0
      if (ielim.le.n)idn(n+1)=1
40      return
        end
c       this subroutine calculates the sum of the squares
c       of the differences of the eigenvectors for a test
c       of convergence
c       description of parameters
c          n     - number of occupied orbitals in this spin state
c          vecto - matrix of old eigenvectors
c          vectn - matrix of current eigenvectors
c          sq    - returned sum of squares of the differences of
```

```
c                 the n occupied orbitals in this spin state
        subroutine sqdif(n,vecto,vectn,work1,work2,sq)
        implicit double precision (a-h,o-z)
        common /files/nbas,ifile,maxbas/a3/a3
        dimension a3(1),vecto(1),vectn(1),work1(1),work2(1)
        sq=0.
        do 20 i=1,n
        ii=(i-1)*nbas
        do 10 j=1,nbas
10      work1(j)=vecto(ii+j)-vectn(ii+j)
        call gdotpr(work1,a3,work1,work2,temp,nbas)
20      sq=sq+temp
        return
        end
c     if istovp=1 is specified, this subroutine sorts the current
c     iteration eigenvectors so that the overlap with the corresponding
c     eigenvector of the previous iteration is a maximum.
c     sorting is done using a row pivoting strategy on the
c     implicitly constructed matrix of overlaps.
c     this subroutine always adjusts the current eigenvectors
c     so that they have the same phase as the corresponding
c     eigenvector of the previous iteration.
      subroutine srtovp(vecto,vectn,values,ispin,istovp)
      implicit double precision (a-h,o-z)
      common /a3/a3(1)/files/nbas,ifile,maxbas
      dimension vecto(1),vectn(1),values(2,1)
      do 100 i=1,nbas
      ii=(i-1)*nbas
      ii1=ii+1
      call gdotpr(vecto(ii1),a3,vectn(ii1),0,tmax,nbas)
      if (istovp.ne.1) goto 50
      imax=i
      i1=i+1
      do 30 j=i1,nbas
      jj=(j-1)*nbas
      jj1=jj+1
      call gdotpr(vecto(ii1),a3,vectn(jj1),0,temp,nbas)
      if (dabs(temp).le.dabs(tmax)) goto 30
      tmax=temp
      imax=j
30    continue
      if (imax.eq.i) goto 50
      iimax=(imax-1)*nbas
      do 40 j=1,nbas
      temp=vectn(ii+j)
      vectn(ii+j)=vectn(iimax+j)
40    vectn(iimax+j)=temp
      i0=i
      temp=values(ispin,i0)
      values(ispin,i0)=values(ispin,imax)
      values(ispin,imax)=temp
50    continue
      if (tmax.gt.0.) goto 100
      do 60 j=1,nbas
60    vectn(ii+j)=-vectn(ii+j)
100   continue
      return
      end
c     this subroutine computes the generalized dot product of vectors
c     x and y with respect to the metric tensor a.
c     description of parameters:
c       x     - left input vector (n)
c       a     - metric tensor (n*n)
c       y     - right input vector (n)
```

```
c         work  - scratch vector (n)
c                 (not needed this version, specify as 0)
c         value - returned value of the dot product.
c         n     - length of each vector.
c       x and y need not be distinct.
        subroutine gdotpr(x,a,y,work,value,n)
        real*8 x(1),a(1),y(1),work(1),value
        value=0.
        do 100 j=1,n
        jj=(j-1)*n
        do 100 i=1,n
100     value=value+x(i)*a(jj+i)*y(j)
        return
        end
c         this subroutine evaluates the total energy
          subroutine et(etn,nup,ndn)
          implicit double precision (a-h,o-z)
          common/files/nbas,ifile,maxbas
          common enrep/a/a(1)/b/b(1)/e/e(2,1)/iup/iup(1)/idn/
       1  idn(1)/rr/rr(1)/sav/sav(1024)/a1/a1(1)/a2/a2(1)/a3/a3(1)/a4/
       1  a4(1)/a5/a5(1)/a6/a6(1)
          etn=0.0
          do 1 i=1,nup
    1     etn=etn+e(1,i)
          if(ndn.eq.0) go to 2
          do 3 i=1,ndn
    3     etn=etn+e(2,i)
    2     continue
          do 14 i=1,nup
          m15=(i-1)*nbas+1
          call gdotpr(a1(m15),a4,a1(m15),0,s,nbas)
   14     etn=etn+s
          if(ndn.eq.0)go to 16
          do 17 i=1,ndn
          m15=(i-1)*nbas+1
          call gdotpr(a2(m15),a4,a2(m15),0,s,nbas)
   17     etn=etn+s
   16     etn=etn/2.0+enrep
          return
          end
c       this subroutine adds the integrals into the
c       spin-up/spin-down hamiltonians
          subroutine two
          implicit double precision(a-h,o-z)
          integer*2 mu1(1024),ii1(1024),jj1(1024),kk1(1024),ll1(1024)
       1,itg1(1024)
          common/files/nbas,ifile,maxbas
          common/inxcm/isnx(1)
          common enrep,nup,ndn/a/a(1)/b/b(1)/e/e(2,1)/iup/
       1  iup(1)/idn/idn(1)/rr/rr(1)/sav/sav(1024)/hu/hu(1)/
       1  hd/hd(1)/a1/a1(1)/a2/a2(1)/a3/a3(1)/a4/a4(1)/a5/a5(1)/
       1  a6/a6(1)
        dimension saver(1024)
          common/lb/ipak(1024)
c        if using combined integrals-labels tape nints=number of integrals=number of labels
c        if using separate integrals and labels tapes nintl=number of integrals,
c        nints=number of labels
c        rewind and position tapes 3 and 4
          rewind 4
          read(4)label
          read(4)nbfn
          read(4)
          read(4)
          read(4)enrep
```

```
      do 100 ict=1,3
      read(4)nlab
206   read(4)nints,last
      if(last.eq.0)go to 206
100   continue
      if(ifile.ne.1)go to 2001
      rewind 3
      read(4)nlabl
      read(4)nint1,last1,saver
      read(3)
      read(3)
      do 200 i=1,3
      read(3)
 201  read(3)ilab,ifml
      if(ifml.eq.0)go to 201
 200  continue
2001  continue
c    put up spin rho into a, down spin rho into b
      if(ndn.eq.0)go to 5
      do 6 i=1,nbas
      do 6 j=1,nbas
      k21=(i-1)*nbas+j
      a(k21)=0.0
      b(k21)=0.0
      do 4 k=1,nup
4     a(k21)=a(k21)+a1((k-1)*nbas+i)*a1((k-1)*nbas+j)
      do 6 k=1,ndn
6     b(k21)=b(k21)+a2((k-1)*nbas+i)*a2((k-1)*nbas+j)
      go to 9
c    case of no down spin electrons
5     do 40 i=1,nbas
      do 40 j=1,nbas
      k21=(i-1)*nbas+j
      a(k21)=0.0
      do 40 k=1,nup
40    a(k21)=a(k21)+a1((k-1)*nbas+i)*a1((k-1)*nbas+j)
    9 k=(maxbas+1)*maxbas/2
      do 1 i=1,k
      hu(i)=0.0
    1 hd(i)=0.0
    8 if(ndn.ne.0)go to 2
      k=maxbas*maxbas
      do 7 i=1,k
    7 b(i)=0.0
    2 continue
      intwh=0
      if(ifile.ne.1)read(4)nlab
    if (ifile.ne.0) read(3)nlab
  72 if (ifile.ne.0) read(3)nints,last,iil,jj1,kk1,ll1,itg1,mu1
      if(ifile.ne.1)read(4)nints,last,iil,jj1,kk1,ll1,itg1,mu1,saver
      if(nints.eq.0)go to 30
      do 29 m1=1,nints
    i=iil(m1)
    j=jj1(m1)
    k=kk1(m1)
    l=ll1(m1)
    itag=itg1(m1)
    mu=mu1(m1)
      if(itag-1)2017,2013,2015
c     old integral
2013  aint=xint
      go to 2031
c     negative of old integral
2015  aint=-xint
```

```
        go to 2031
c       new integral required
2017    if(ifile)2018,2020,2018
2020    intwh=m1
        xint=saver(m1)
        aint=xint
        go to 2031
2018    intwh=intwh+1
        if(intwh-nint1)2021,2021,2019
2021    xint=saver(intwh)
        aint=xint
        go to 2031
2019    if(last1.ne.0)stop'not enough unique integrals/
     1  too many labels'
        intwh=1
        read(4)nint1,last1,saver
        xint=saver(1)
        aint=xint
 2031  continue
      taint=aint+aint
      if(mu.lt.1.or.mu.gt.14) go to 1999
      mu=15-mu
      go to (25,24,23,22,21,20,19,18,17,16,15,14,13,12),mu
c     iiii
   12 ii=isnx(i)+i
      nii=(i-1)*nbas+i
      hu(ii)=hu(ii)+aint*b(nii)
      hd(ii)=hd(ii)+aint*a(nii)
      go to 29
c     ijij
   13 ij=isnx(i)+j
      ii=isnx(i)+i
      jj=isnx(j)+j
      nij=(i-1)*nbas+j
      njj=(j-1)*nbas+j
      nii=(i-1)*nbas+i
      hu(ij)=hu(ij)+aint*(a(nij)+2.0*b(nij))
      hu(ii)=hu(ii)-aint*a(njj)
      hu(jj)=hu(jj)-aint*a(nii)
      hd(ij)=hd(ij)+aint*(b(nij)+2.0*a(nij))
      hd(ii)=hd(ii)-aint*b(njj)
      hd(jj)=hd(jj)-aint*b(nii)
      go to 29
c     iijj
   14 ii=isnx(i)+i
      kk=isnx(k)+k
      ik=isnx(i)+k
      nkk=(k-1)*nbas+k
      nii=(i-1)*nbas+i
      nik=(i-1)*nbas+k
      hu(ii)=hu(ii)+aint*(a(nkk)+b(nkk))
      hd(ii)=hd(ii)+aint*(a(nkk)+b(nkk))
      hu(kk)=hu(kk)+aint*(a(nii)+b(nii))
      hd(kk)=hd(kk)+aint*(a(nii)+b(nii))
      hu(ik)=hu(ik)-aint*a(nik)
      hd(ik)=hd(ik)-aint*b(nik)
      go to 29
c     iiil
   15 il=isnx(i)+l
      ii=isnx(i)+i
      nii=(i-1)*nbas+i
      nil=(i-1)*nbas+l
      hu(il)=hu(il)+aint*b(nii)
      hd(il)=hd(il)+aint*a(nii)
```

```
      hu(ii)=hu(ii)+taint*b(nil)
      hd(ii)=hd(ii)+taint*a(nil)
      go to 29
c     iikl
   16 kl=isnx(k)+l
      ii=isnx(i)+i
      il=isnx(i)+l
      ik=isnx(i)+k
      nii=(i-1)*nbas+i
      nkl=(k-1)*nbas+l
      nik=(i-1)*nbas+k
      nil=(i-1)*nbas+l
      hu(kl)=hu(kl)+aint*(a(nii)+b(nii))
      hu(ii)=hu(ii)+taint*(a(nkl)+b(nkl))
      hu(il)=hu(il)-aint*a(nik)
      hu(ik)=hu(ik)-aint*a(nil)
      hd(kl)=hd(kl)+aint*(b(nii)+a(nii))
      hd(ii)=hd(ii)+taint*(b(nkl)+a(nkl))
      hd(il)=hd(il)-aint*b(nik)
      hd(ik)=hd(ik)-aint*b(nil)
      go to 29
c     ijjj
   17 jj=isnx(j)+j
      ij=isnx(i)+j
      nij=(i-1)*nbas+j
      njj=(j-1)*nbas+j
      hu(jj)=hu(jj)+taint*b(nij)
      hd(jj)=hd(jj)+taint*a(nij)
      hu(ij)=hu(ij)+aint*b(njj)
      hd(ij)=hd(ij)+aint*a(njj)
      go to 29
c     ijkk,j gt k
   18 kk=isnx(k)+k
      ij=isnx(i)+j
      jk=isnx(j)+k
      ik=isnx(i)+k
      nij=(i-1)*nbas+j
      nkk=(k-1)*nbas+k
      njk=(j-1)*nbas+k
      nik=(i-1)*nbas+k
      hu(kk)=hu(kk)+taint*(a(nij)+b(nij))
      hu(ij)=hu(ij)+aint*(a(nkk)+b(nkk))
      hu(ik)=hu(ik)-aint*a(njk)
      hu(jk)=hu(jk)-aint*a(nik)
      hd(kk)=hd(kk)+taint*(b(nij)+a(nij))
      hd(ij)=hd(ij)+aint*(b(nkk)+a(nkk))
      hd(ik)=hd(ik)-aint*b(njk)
      hd(jk)=hd(jk)-aint*b(nik)
      go to 29
c     ijkk  j lt k
   19 kk=isnx(k)+k
      ij=isnx(i)+j
      jk=isnx(k)+j
      ik=isnx(i)+k
      nij=(i-1)*nbas+j
      nkk=(k-1)*nbas+k
      njk=(j-1)*nbas+k
      nik=(i-1)*nbas+k
      hu(kk)=hu(kk)+taint*(a(nij)+b(nij))
      hu(ij)=hu(ij)+aint*(a(nkk)+b(nkk))
      hu(ik)=hu(ik)-aint*a(njk)
      hu(jk)=hu(jk)-aint*a(nik)
      hd(kk)=hd(kk)+taint*(b(nij)+a(nij))
      hd(ij)=hd(ij)+aint*(b(nkk)+a(nkk))
```

```
      hd(ik)=hd(ik)-aint*b(njk)
      hd(jk)=hd(jk)-aint*b(nik)
      go to 29
c     ijjl
   20 ij=isnx(i)+j
      jl=isnx(j)+l
      jj=isnx(j)+j
      il=isnx(i)+l
      njl=(j-1)*nbas+l
      nij=(i-1)*nbas+j
      nil=(i-1)*nbas+l
      njj=(j-1)*nbas+j
      hu(ij)=hu(ij)+aint*(a(njl)+2.0*b(njl))
      hu(jl)=hu(jl)+aint*(a(nij)+2.0*b(nij))
      hu(jj)=hu(jj)-taint*a(nil)
      hu(il)=hu(il)-aint*a(njj)
      hd(ij)=hd(ij)+aint*(b(njl)+2.0*a(njl))
      hd(jl)=hd(jl)+aint*(b(nij)+2.0*a(nij))
      hd(jj)=hd(jj)-taint*b(nil)
      hd(il)=hd(il)-aint*b(njj)
      go to 29
c     ijil
   21 il=isnx(i)+l
      ij=isnx(i)+j
      ii=isnx(i)+i
      jl=isnx(j)+l
      nil=(i-1)*nbas+l
      nij=(i-1)*nbas+j
      njl=(j-1)*nbas+l
      nii=(i-1)*nbas+i
      hu(ij)=hu(ij)+aint*(a(nil)+2.0*b(nil))
      hu(il)=hu(il)+aint*(a(nij)+2.0*b(nij))
      hu(ii)=hu(ii)-taint*a(njl)
      hu(jl)=hu(jl)-aint*a(nii)
      hd(ij)=hd(ij)+aint*(b(nil)+2.0*a(nil))
      hd(il)=hd(il)+aint*(b(nij)+2.0*a(nij))
      hd(ii)=hd(ii)-taint*b(njl)
      hd(jl)=hd(jl)-aint*b(nii)
      go to 29
c     ijkj
   22 ij=isnx(i)+j
      kj=isnx(k)+j
      jj=isnx(j)+j
      ik=isnx(i)+k
      nkj=(k-1)*nbas+j
      nij=(i-1)*nbas+j
      nik=(i-1)*nbas+k
      njj=(j-1)*nbas+j
      hu(ij)=hu(ij)+aint*(a(nkj)+2.0*b(nkj))
      hu(kj)=hu(kj)+aint*(a(nij)+2.0*b(nij))
      hu(jj)=hu(jj)-taint*a(nik)
      hu(ik)=hu(ik)-aint*a(njj)
      hd(ij)=hd(ij)+aint*(b(nkj)+2.0*a(nkj))
      hd(kj)=hd(kj)+aint*(b(nij)+2.0*a(nij))
      hd(jj)=hd(jj)-taint*b(nik)
      hd(ik)=hd(ik)-aint*b(njj)
      go to 29
c     ijkl  j.gt.k,j.gt.l
   23 kl=isnx(k)+l
      jk=isnx(j)+k
      jl=isnx(j)+l
      ij=isnx(i)+j
      il=isnx(i)+l
      ik=isnx(i)+k
```

```
      nkl=(k-1)*nbas+l
      nij=(i-1)*nbas+j
      njl=(j-1)*nbas+l
      nik=(i-1)*nbas+k
      nil=(i-1)*nbas+l
      njk=(j-1)*nbas+k
      hu(ij)=hu(ij)+taint*(a(nkl)+b(nkl))
      hu(kl)=hu(kl)+taint*(a(nij)+b(nij))
      hu(ik)=hu(ik)-aint*a(njl)
      hu(jl)=hu(jl)-aint*a(nik)
      hu(jk)=hu(jk)-aint*a(nil)
      hu(il)=hu(il)-aint*a(njk)
      hd(ij)=hd(ij)+taint*(a(nkl)+b(nkl))
      hd(kl)=hd(kl)+taint*(a(nij)+b(nij))
      hd(ik)=hd(ik)-aint*b(njl)
      hd(jl)=hd(jl)-aint*b(nik)
      hd(jk)=hd(jk)-aint*b(nil)
      hd(il)=hd(il)-aint*b(njk)
      go to 29
c     ijkl  j.lt.k,j.gt.l
   24 kl=isnx(k)+l
      jk=isnx(k)+j
      jl=isnx(j)+l
      ij=isnx(i)+j
      il=isnx(i)+l
      ik=isnx(i)+k
      nkl=(k-1)*nbas+l
      nij=(i-1)*nbas+j
      njl=(j-1)*nbas+l
      nik=(i-1)*nbas+k
      nil=(i-1)*nbas+l
      njk=(j-1)*nbas+k
      hu(ij)=hu(ij)+taint*(a(nkl)+b(nkl))
      hu(kl)=hu(kl)+taint*(a(nij)+b(nij))
      hu(ik)=hu(ik)-aint*a(njl)
      hu(jl)=hu(jl)-aint*a(nik)
      hu(jk)=hu(jk)-aint*a(nil)
      hu(il)=hu(il)-aint*a(njk)
      hd(ij)=hd(ij)+taint*(a(nkl)+b(nkl))
      hd(kl)=hd(kl)+taint*(a(nij)+b(nij))
      hd(ik)=hd(ik)-aint*b(njl)
      hd(jl)=hd(jl)-aint*b(nik)
      hd(jk)=hd(jk)-aint*b(nil)
      hd(il)=hd(il)-aint*b(njk)
      go to 29
c     ijkl  j.lt.k,j.lt.l
   25 kl=isnx(k)+l
      jk=isnx(k)+j
      jl=isnx(l)+j
      ij=isnx(i)+j
      il=isnx(i)+l
      ik=isnx(i)+k
      nkl=(k-1)*nbas+l
      nij=(i-1)*nbas+j
      njl=(j-1)*nbas+l
      nik=(i-1)*nbas+k
      nil=(i-1)*nbas+l
      njk=(j-1)*nbas+k
      hu(ij)=hu(ij)+taint*(a(nkl)+b(nkl))
      hu(kl)=hu(kl)+taint*(a(nij)+b(nij))
      hu(ik)=hu(ik)-aint*a(njl)
      hu(jl)=hu(jl)-aint*a(nik)
      hu(jk)=hu(jk)-aint*a(nil)
      hu(il)=hu(il)-aint*a(njk)
```

```
      hd(ij)=hd(ij)+taint*(a(nkl)+b(nkl))
      hd(kl)=hd(kl)+taint*(a(nij)+b(nij))
      hd(ik)=hd(ik)-aint*b(njl)
      hd(jl)=hd(jl)-aint*b(nik)
      hd(jk)=hd(jk)-aint*b(nil)
      hd(il)=hd(il)-aint*b(njk)
29    continue
30      if(last.eq.0)go to 72
        do 31 i=1,nbas
        do 31 j=1,i
        indh=isnx(i)+j
        k6=(j-1)*nbas+i
        k7=(i-1)*nbas+j
        a5(k6)=hu(indh)
        a5(k7)=hu(indh)
        a6(k6)=hd(indh)
31      a6(k7)=hd(indh)
        do 33 i=1,nbas
        do 33 j=1,nbas
        k6=(i-1)*nbas+j
      a5(k6)=a5(k6)+a4(k6)
      a6(k6)=a6(k6)+a4(k6)
  33  continue
        return
 1999 write(60,502)mu
  502 format(' mu out of range in subroutine two  mu =',i4)
        end
c
c     .................................................................
c
c        subroutine xroot -- this subroutine is a modified version of nroot
c
c        purpose
c           compute eigenvalues and eigenvectors of a real nonsymmetric
c           matrix of the form b-inverse times a.  this subroutine is
c           normally called by subroutine canor in performing a
c           canonical correlation analysis.
c
c        usage
c           call xroot (m,a,bl,bx,xl,x,work1,work2)
c
c        description of parameters
c           m     - order of square matrices a, b, bx, and x.
c           a     - input matrix (m x m) (full matrix, destroyed).
c           bl    - input vector of length m containing eigenvalues
c                   of b.
c           bx    - input matrix (m x m) containing eigenvectors
c                   of b (destroyed).
c           xl    - output vector of length m containing eigenvalues of
c                   b-inverse times a.
c           x     - output matrix (m x m) containing eigenvectors column-
c                   wise of b-inverse times a.
c           work1- work vector of length m.
c           work2- work vector of length m.
c
c        remarks
c           note that the matrix b is assumed to be positive
c           definite, that is, each of its eigenvalues must be
c           positive.
c           note also that b is never passed to this routine,
c           only its eigenvalues and eigenvectors are needed.
c
c        subroutines and function subprograms required
c           eispack path rsp
```

```
c
c       method
c          refer to w. w. cooley and p. r. lohnes, 'multivariate pro-
c          cedures for the behavioral sciences', john wiley and sons,
c          1962, chapter 3.
c
c     .................................................................
c
      subroutine xroot (m,a,bl,bx,xl,x,work1,work2)
      dimension a(1),bl(1),bx(1),xl(1),x(1),work1(1),work2(1)
c
c        ..............................................................
c
c        if a single precision version of this routine is desired, a
c        c should be placed in column 1 of the double precision
c        statement which follows.
c
      double precision a,bl,bx,xl,x,sumv,work1,work2
c
c        the c must also be placed in double precision statements
c        appearing in other routines used in conjunction with this
c        routine.
c
c        the single precision version of this subroutine must also
c        contain single precision fortran functions.dsqrt in statements
c        110 and 175 must be changed to  sqrt.dabs in statement 110
c        must be changed to  abs.
c
c        ..............................................................
c
c
c     form reciprocals of square root of eigenvalues.  the results
c     are premultiplied by the associated eigenvectors.
c
      do 110 j=1,m
  110 xl(j)=1.0/dsqrt(dabs(bl(j)))
      k=0
      do 115 j=1,m
      do 115 i=1,m
      k=k+1
  115 x(k)=bx(k)*xl(j)
c
c     form (b**(-1/2))prime * a * (b**(-1/2))
c
      do 120 i=1,m
      n2=0
      do 120 j=1,m
      n1=m*(i-1)
      l=m*(j-1)+i
      bx(l)=0.0
      do 120 k=1,m
      n1=n1+1
      n2=n2+1
  120 bx(l)=bx(l)+x(n1)*a(n2)
      l=0
      do 130 j=1,m
      do 130 i=1,j
      n1=i-m
      n2=m*(j-1)
      l=l+1
      a(l)=0.0
      do 130 k=1,m
      n1=n1+m
      n2=n2+1
```

```
  130 a(l)=a(l)+bx(n1)*x(n2)
c
c     compute eigenvalues and eigenvectors of a
c
      ma=m*(m+1)/2
      call rsp(m,m,ma,a,xl,1,bx,work1,work2,ierr)
      if (ierr.ne.0) stop 'trouble with the eigenvalue finder.'
c
c     compute the normalized eigenvectors
c
      do 150 i=1,m
      n2=0
      do 150 j=1,m
      n1=i-m
      l=m*(j-1)+i
      a(l)=0.0
      do 150 k=1,m
      n1=n1+m
      n2=n2+1
  150 a(l)=a(l)+x(n1)*bx(n2)
      l=0
      k=0
      do 180 j=1,m
      sumv=0.0
      do 170 i=1,m
      l=l+1
  170 sumv=sumv+a(l)*a(l)
  175 sumv=dsqrt(sumv)
      do 180 i=1,m
      k=k+1
  180 x(k)=a(k)/sumv
      return
      end
c
c     -----------------------------------------------------------------
c
      subroutine rsp(nm,n,nv,a,w,matz,z,fv1,fv2,ierr)
c
      integer i,j,n,nm,nv,ierr,matz
      double precision a(nv),w(n),z(nm,n),fv1(n),fv2(n)
c
c     this subroutine calls the recommended sequence of
c     subroutines from the eigensystem subroutine package (eispack)
c     to find the eigenvalues and eigenvectors (if desired)
c     of a real symmetric packed matrix.
c
c     on input-
c
c        nm  must be set to the row dimension of the two-dimensional
c        array parameters as declared in the calling program
c        dimension statement,
c
c        n  is the order of the matrix  a,
c
c        nv  is an integer variable set equal to the
c        dimension of the array  a  as specified for
c        a  in the calling program.  nv  must not be
c        less than  n*(n+1)/2,
c
c        a  contains the lower triangle of the real symmetric
c        packed matrix stored row-wise,
c
c        matz  is an integer variable set equal to zero if
c        only eigenvalues are desired,  otherwise it is set to
```

```
c         any non-zero integer for both eigenvalues and eigenvectors.
c
c      on output-
c
c         w  contains the eigenvalues in ascending order,
c
c         z  contains the eigenvectors if matz is not zero,
c
c         ierr  is an integer output variable set equal to an
c         error completion code described in section 2b of the
c         documentation.  the normal completion code is zero,
c
c         fv1  and  fv2  are temporary storage arrays.
c
c      questions and comments should be directed to b. s. garbow,
c      applied mathematics division, argonne national laboratory
c
c      ------------------------------------------------------------------
c
      if (n .le. nm) go to 5
      ierr = 10 * n
      go to 50
    5 if (nv .ge. (n * (n + 1)) / 2) go to 10
      ierr = 20 * n
      go to 50
c
   10 call  tred3(n,nv,a,w,fv1,fv2)
      if (matz .ne. 0) go to 20
c     ********** don't find eigenvalues only **********
c     call  tqlrat(n,w,fv2,ierr)
      stop
c     ********** find both eigenvalues and eigenvectors **********
   20 do 40 i = 1, n
c
         do 30 j = 1, n
            z(j,i) = 0.0
   30    continue
c
         z(i,i) = 1.0
   40 continue
c
      call  tql2(nm,n,w,fv1,z,ierr)
      if (ierr .ne. 0) go to 50
      call  trbak3(nm,n,nv,a,n,z)
   50 return
c     ********** last card of rsp **********
      end
c
c     ------------------------------------------------------------------
c
      subroutine tred3(n,nv,a,d,e,e2)
c
      integer i,j,k,l,n,ii,iz,jk,nv
      double precision a(nv),d(n),e(n),e2(n)
      double precision f,g,h,hh,scale
c     real sqrt,abs,sign
c
c     this subroutine is a translation of the algol procedure tred3,
c     num. math. 11, 181-195(1968) by martin, reinsch, and wilkinson.
c     handbook for auto. comp., vol.ii-linear algebra, 212-226(1971).
c
c     this subroutine reduces a real symmetric matrix, stored as
c     a one-dimensional array, to a symmetric tridiagonal matrix
c     using orthogonal similarity transformations.
```

```
c
c     on input-
c
c        n is the order of the matrix,
c
c        nv must be set to the dimension of the array parameter a
c          as declared in the calling program dimension statement,
c
c        a contains the lower triangle of the real symmetric
c          input matrix, stored row-wise as a one-dimensional
c          array, in its first n*(n+1)/2 positions.
c
c     on output-
c
c        a contains information about the orthogonal
c          transformations used in the reduction,
c
c        d contains the diagonal elements of the tridiagonal matrix,
c
c        e contains the subdiagonal elements of the tridiagonal
c          matrix in its last n-1 positions.  e(1) is set to zero,
c
c        e2 contains the squares of the corresponding elements of e.
c          e2 may coincide with e if the squares are not needed.
c
c     questions and comments should be directed to b. s. garbow,
c     applied mathematics division, argonne national laboratory
c
c     ------------------------------------------------------------------
c
c     ********** for i=n step -1 until 1 do -- **********
      do  300 ii = 1, n
         i = n + 1 - ii
         l = i - 1
         iz = (i * l) / 2
         h = 0.0
         scale = 0.0
         if (l .lt. 1) go to 130
c     ********** scale row (algol tol then not needed) **********
         do 120 k = 1, l
            iz = iz + 1
            d(k) = a(iz)
            scale = scale + dabs(d(k))
  120    continue
c
         if (scale .ne. 0.0) go to 140
  130    e(i) = 0.0
         e2(i) = 0.0
         go to 290
c
  140    do 150 k = 1, l
            d(k) = d(k) / scale
            h = h + d(k) * d(k)
  150    continue
c
         e2(i) = scale * scale * h
         f = d(l)
         g = -dsign(dsqrt(h),f)
         e(i) = scale * g
         h = h - f * g
         d(l) = f - g
         a(iz) = scale * d(l)
         if (l .eq. 1) go to 290
         f = 0.0
```

```
c                                                                       28445082
         do 240 j = 1, l                                                28445083
            g = 0.0                                                     28445084
            jk = (j * (j-1)) / 2                                        28445085
c     ********** form element of a*u **********                         28445086
            do 180 k = 1, l                                             28445087
               jk = jk + 1                                              28445088
               if (k .gt. j) jk = jk + k - 2                            28445089
               g = g + a(jk) * d(k)                                     28445090
  180       continue                                                    28445091
c     ********** form element of p **********                           28445092
            e(j) = g / h                                                28445093
            f = f + e(j) * d(j)                                         28445094
  240    continue                                                       28445095
c                                                                       28445096
         hh = f / (h + h)                                               28445097
         jk = 0                                                         28445098
c     ********** form reduced a **********                              28445099
         do 260 j = 1, l                                                28445100
            f = d(j)                                                    28445101
            g = e(j) - hh * f                                           28445102
            e(j) = g                                                    28445103
c                                                                       28445104
            do 260 k = 1, j                                             28445105
               jk = jk + 1                                              28445106
               a(jk) = a(jk) - f * e(k) - g * d(k)                      28445107
  260    continue                                                       28445108
c                                                                       28445109
  290    d(i) = a(iz+1)                                                 28445110
         a(iz+1) = scale * dsqrt(h)                                     28445111
  300 continue                                                          28445112
c                                                                       28445113
      return                                                            28445114
c     ********** last card of tred3 **********                          28445115
      end                                                               28445116
c                                                                      902210241
c     -----------------------------------------------------------------902210242
c                                                                      902210243
      subroutine tql2(nm,n,d,e,z,ierr)                                 902210244
c                                                                      902210245
      integer i,j,k,l,m,n,ii,l1,nm,mml,ierr                            902210246
      double precision d(n),e(n),z(nm,n)                               902210247
      double precision b,c,f,g,h,p,r,s,machep                          902210248
c     real sqrt,abs,sign                                               902210249
c                                                                       90225010
c     this subroutine is a translation of the algol procedure tql2,     90225011
c     num. math. 11, 293-306(1968) by bowdler, martin, reinsch, and     90225012
c     wilkinson.                                                        90225013
c     handbook for auto. comp., vol.ii-linear algebra, 227-240(1971).   90225014
c                                                                       90225015
c     this subroutine finds the eigenvalues and eigenvectors            90225016
c     of a symmetric tridiagonal matrix by the ql method.               90225017
c     the eigenvectors of a full symmetric matrix can also              90225018
c     be found if  tred2  has been used to reduce this                  90225019
c     full matrix to tridiagonal form.                                  90225020
c                                                                       90225021
c     on input-                                                         90225022
c                                                                       90225023
c        nm must be set to the row dimension of two-dimensional         90225024
c          array parameters as declared in the calling program          90225025
c          dimension statement,                                         90225026
c                                                                       90225027
c        n is the order of the matrix,                                  90225028
c                                                                       90225029
```

```
c        d contains the diagonal elements of the input matrix,
c
c        e contains the subdiagonal elements of the input matrix
c          in its last n-1 positions.  e(1) is arbitrary,
c
c        z contains the transformation matrix produced in the
c          reduction by  tred2, if performed.  if the eigenvectors
c          of the tridiagonal matrix are desired, z must contain
c          the identity matrix.
c
c      on output-
c
c        d contains the eigenvalues in ascending order.  if an
c          error exit is made, the eigenvalues are correct but
c          unordered for indices 1,2,...,ierr-1,
c
c        e has been destroyed,
c
c        z contains orthonormal eigenvectors of the symmetric
c          tridiagonal (or full) matrix.  if an error exit is made,
c          z contains the eigenvectors associated with the stored
c          eigenvalues,
c
c        ierr is set to
c          zero       for normal return,
c          j          if the j-th eigenvalue has not been
c                     determined after 30 iterations.
c
c     questions and comments should be directed to b. s. garbow,
c     applied mathematics division, argonne national laboratory
c
c     -----------------------------------------------------------------
c
c     ********** machep is a machine dependent parameter specifying
c                the relative precision of floating point arithmetic.
c
c                **********
      machep = 2.d0**(-53)
c
      ierr = 0
      if (n .eq. 1) go to 1001
c
      do 100 i = 2, n
  100 e(i-1) = e(i)
c
      f = 0.0
      b = 0.0
      e(n) = 0.0
c
      do 240 l = 1, n
         j = 0
         h = machep * (dabs(d(l)) + dabs(e(l)))
         if (b .lt. h) b = h
c     ********** look for small sub-diagonal element **********
         do 110 m = l, n
            if (abs(e(m)) .le. b) go to 120
c     ********** e(n) is always zero, so there is no exit
c                through the bottom of the loop **********
  110    continue
c
  120    if (m .eq. l) go to 220
  130    if (j .eq. 30) go to 1000
         j = j + 1
c     ********** form shift **********
```

```
         l1 = l + 1
         g = d(l)
         p = (d(l1) - g) / (2.0 * e(l))
         r = dsqrt(p*p+1.0)
         d(l) = e(l) / (p + dsign(r,p))
         h = g - d(l)
c
         do 140 i = l1, n
  140    d(i) = d(i) - h
c
         f = f + h
c     ********** ql transformation **********
         p = d(m)
         c = 1.0
         s = 0.0
         mml = m - 1
c     ********** for i=m-1 step -1 until l do -- **********
         do 200 ii = 1, mml
            i = m - ii
            g = c * e(i)
            h = c * p
            if (dabs(p) .lt. dabs(e(i))) go to 150
            c = e(i) / p
            r = dsqrt(c*c+1.0)
            e(i+1) = s * p * r
            s = c / r
            c = 1.0 / r
            go to 160
  150       c = p / e(i)
            r = sqrt(c*c+1.0)
            e(i+1) = s * e(i) * r
            s = 1.0 / r
            c = c * s
  160       p = c * d(i) - s * g
            d(i+1) = h + s * (c * g + s * d(i))
c     ********** form vector **********
            do 180 k = 1, n
               h = z(k,i+1)
               z(k,i+1) = s * z(k,i) + c * h
               z(k,i) = c * z(k,i) - s * h
  180       continue
c
  200    continue
c
         e(l) = s * p
         d(l) = c * p
         if (dabs(e(l)) .gt. b) go to 130
  220    d(l) = d(l) + f
  240 continue
c     ********** order eigenvalues and eigenvectors **********
      do 300 ii = 2, n
         i = ii - 1
         k = i
         p = d(i)
c
         do 260 j = ii, n
            if (d(j) .ge. p) go to 260
            k = j
            p = d(j)
  260    continue
c
         if (k .eq. i) go to 300
         d(k) = d(i)
         d(i) = p
```

```
c
         do 280 j = 1, n
            p = z(j,i)
            z(j,i) = z(j,k)
            z(j,k) = p
  280    continue
c
  300 continue
c
      go to 1001
c     ********** set error -- no convergence to an
c                eigenvalue after 30 iterations **********
 1000 ierr = 1
 1001 return
c     ********** last card of tql2 **********
      end
c
c     ----------------------------------------------------------------
c
      subroutine trbak3(nm,n,nv,a,m,z)
c
      integer i,j,k,l,m,n,ik,iz,nm,nv
      double precision a(nv),z(nm,m)
      double precision h,s
c
c     this subroutine is a translation of the algol procedure trbak3,
c     num. math. 11, 181-195(1968) by martin, reinsch, and wilkinson.
c     handbook for auto. comp., vol.ii-linear algebra, 212-226(1971).
c
c     this subroutine forms the eigenvectors of a real symmetric
c     matrix by back transforming those of the corresponding
c     symmetric tridiagonal matrix determined by  tred3.
c
c     on input-
c
c        nm must be set to the row dimension of two-dimensional
c          array parameters as declared in the calling program
c          dimension statement,
c
c        n is the order of the matrix,
c
c        nv must be set to the dimension of the array parameter a
c          as declared in the calling program dimension statement,
c
c        a contains information about the orthogonal transformations
c          used in the reduction by  tred3  in its first
c          n*(n+1)/2 positions,
c
c        m is the number of eigenvectors to be back transformed,
c
c        z contains the eigenvectors to be back transformed
c          in its first m columns.
c
c     on output-
c
c        z contains the transformed eigenvectors
c          in its first m columns.
c
c     note that trbak3 preserves vector euclidean norms.
c
c     questions and comments should be directed to b. s. garbow,
c     applied mathematics division, argonne national laboratory
c
c     ----------------------------------------------------------------
```

```
c
      if (m .eq. 0) go to 200
      if (n .eq. 1) go to 200
c
      do 140 i = 2, n
         l = i - 1
         iz = (i * l) / 2
         ik = iz + i
         h = a(ik)
         if (h .eq. 0.0) go to 140
c
         do 130 j = 1, m
            s = 0.0
            ik = iz
c
            do 110 k = 1, l
               ik = ik + 1
               s = s + a(ik) * z(k,j)
  110       continue
c     ********** double division avoids possible underflow **********
            s = (s / h) / h
            ik = iz
c
            do 120 k = 1, l
               ik = ik + 1
 100   z(k,j)=z(k,j)-s*a(ik)
  120       continue
c
  130    continue
c
  140 continue
c
  200 return
c     ********** last card of trbak3 **********
      end
      subroutine lister(ino,jno)
c     polylbls program -- mqm master file log
c     initial creation -- 4/8/74 -- bdo
c     cit integral list generation program-based on polyatom pa20
      integer pkdlbl
      character*16 naml
      character*11 mol12,mol05,mol02
      dimension ma(180,50),mb(180,50),ia(180),ib(180),nam(4),ilbl(20)
     1,mol12(20),mol05(20),mol02(20)
      real*8 tymes,xtyme(2)
      dimension tyme(2)
      common/ndata/nbfn,nbfo,ntrn,ntrnpt,nadd,ntape
      common/ioind/icon(10),mrec,izero,ione
      common/label/pkdlbl(1024)
      data nam/4hslst,4htlst,4hvlst,4hmlst/,mxbf/180/,mxtr/50/
      izero=0
      ione=1
      mrec=1024
      ntape=3
c
c     ***********************************************************************
c
      mol12(1)='mol1201.dat'
      mol12(2)='mol1202.dat'
      mol12(3)='mol1203.dat'
      mol12(4)='mol1204.dat'
      mol12(5)='mol1205.dat'
      mol12(6)='mol1206.dat'
      mol12(7)='mol1207.dat'
```

```
      mol12(8)='mol1208.dat'
      mol12(9)='mol1209.dat'
      mol12(10)='mol1210.dat'
      mol12(11)='mol1211.dat'
      mol12(12)='mol1212.dat'
      mol12(13)='mol1213.dat'
      mol12(14)='mol1214.dat'
      mol12(15)='mol1215.dat'
      mol12(16)='mol1216.dat'
      mol12(17)='mol1217.dat'
      mol12(18)='mol1218.dat'
      mol12(19)='mol1219.dat'
      mol12(20)='mol1220.dat'
      mol02(1)='mol0201.dat'
      mol02(2)='mol0202.dat'
      mol02(3)='mol0203.dat'
      mol02(4)='mol0204.dat'
      mol02(5)='mol0205.dat'
      mol02(6)='mol0206.dat'
      mol02(7)='mol0207.dat'
      mol02(8)='mol0208.dat'
      mol02(9)='mol0209.dat'
      mol02(10)='mol0210.dat'
      mol02(11)='mol0211.dat'
      mol02(12)='mol0212.dat'
      mol02(13)='mol0213.dat'
      mol02(14)='mol0214.dat'
      mol02(15)='mol0215.dat'
      mol02(16)='mol0216.dat'
      mol02(17)='mol0217.dat'
      mol02(18)='mol0218.dat'
      mol02(19)='mol0219.dat'
      mol02(20)='mol0220.dat'
      mol05(1)='mol0501.dat'
      mol05(2)='mol0505.dat'
      mol05(3)='mol0503.dat'
      mol05(4)='mol0504.dat'
      mol05(5)='mol0505.dat'
      mol05(6)='mol0506.dat'
      mol05(7)='mol0507.dat'
      mol05(8)='mol0508.dat'
      mol05(9)='mol0509.dat'
      mol05(10)='mol0510.dat'
      mol05(11)='mol0511.dat'
      mol05(12)='mol0512.dat'
      mol05(13)='mol0513.dat'
      mol05(14)='mol0514.dat'
      mol05(15)='mol0515.dat'
      mol05(16)='mol0516.dat'
      mol05(17)='mol0517.dat'
      mol05(18)='mol0518.dat'
      mol05(19)='mol0519.dat'
      mol05(20)='mol0520.dat'
      open(unit=5,file=mol12(ino),form='formatted')
      open(unit=60,file=mol05(ino),form='formatted')
      open(unit=ntape,file=mol02(ino),
     1form='unformatted')
c     read in input data for this run
      call inlab(ma,mb,ia,ib,mxbf,mxtr,ilbl)
c     the matrix of transformation properties from this routine is ready
c     to use at this point
      rewind ntape
      write(ntape) ilbl
      write(ntape) nbfn
```

```
c     write(ntape) nbfn,ntrn,ntrnpt,((ma(i,j),j=1,ntrn),i=1,nbfn)
      ndo=ntrn
      do 60 k=1,3
      write(ntape) nam(k)
      if(k.eq.3) ndo=ntrnpt
c     get list for one electron integrals-use only molecular point group
c     for the potential energy integral list
   60 call olist(ma,mb,ia,ib,mxbf,mxtr,ndo)
      write(ntape) nam(4)
      if(icon(1).ne.0) go to 75
c     the two electron integral list is now produced
      call tlist(ma,mb,ia,ib,mxbf,mxtr)
   75 endfile ntape
      rewind ntape
1000  format(1x,'enter the name of the labels input file (file 5)')
1100  format(a16)
1200  format(1x,'enter the name of the labels information output file',
     1' (file 6)')
1300  format(1x,'enter the name of the labels output file (file 3)')
      close(unit=ntape)
      close(unit=5)
      close(unit=60)
 1400  format(1x,f18.4,4x,f18.4)
      return
      end
c
c     ********************************************
c
      subroutine inlab(ma,mb,ia,ib,mxbf,mxtr,ilbl)
      dimension ma(mxbf,1),mb(mxbf,1),ia(1),ib(1),ilbl(1)
      dimension ngroup(180)
      common /ndata /  nbfn, nbfo, ntrn, ntrnpt, nadd,ntape
      common /ioind /  icon(10)
      read(5,930) (ilbl(i),i=1,20)
      read  (5,913)  icon
      read  (5,905)  nbfo, nbfn
      read(5,905) (ia(i),i=1,nbfo)
      read(5,905) ntrn,ntrnpt,nadd
      write(60,600)
      write(60,610) (ilbl(i),i=1,20)
      write(60,620) icon
      write(60,630) nbfo,nbfn
      write(60,640) ntrn,ntrnpt,nadd
      k=0
      do 12 i=1,nbfo
      if(ia(i).eq.0) ia(i)=1
   12 k=k+ia(i)
      if(nbfn.le.mxbf.and.nbfo.le.nbfn) go to 701
      write(60,800)
      write(60,801) nbfo,nbfn,mxbf
      stop
  701 if(nbfn.eq.k) go to 702
      write(60,800)
      write(60,802) nbfn,k
      stop
  702 if(ntrn.le.mxtr.and.ntrnpt.le.mxtr) go to 703
      write(60,800)
      write(60,803) ntrn,ntrnpt,mxtr
      stop
  703 continue
      if(ntrn.eq.0) return
      write(60,912) nbfo,ntrn
      do 52 i=1,nbfo
      read(5,905) (ma(i,j),j=1,ntrn)
```

```
      do 350 j=1,ntrn
  350 mb(i,j)=ma(i,j)
   52 write(60,910) (ma(i,j),j=1,ntrn)
c     ********************************************
c     check to see that same number does not occur twice in a column
      if(nbfo.le.1) go to 1740
      do 1741 j=1,ntrn
      do 1742 i=2,nbfo
      if(ma(i,j).eq.0) go to 1742
      itest=iabs(ma(i,j))
      im1=i-1
      do 1743 k=1,im1
      iz=ma(k,j)
      if(iabs(iz).eq.itest) go to 1744
 1743 continue
 1742 continue
 1741 continue
      go to 1740
 1744 continue
      write(60,1750) j,i,k
 1750 format(///5x,7hcolumn  ,i3,22h rows with same class  ,2i5)
      stop
 1740 continue
c
c     ********************************************
c     now the operations read in are multiplied together to produce a
c     group
c     a new operation is checked and added,then multiplied with all old
c     ones
c     all multiplications with the set of operations in the matrix are
c     tried before adding a new element
c     if point group operations precede local symmetry operations-all
c     point group operations will occur before any local sym op-this is
c     program expects it
      if(icon(2).ne.0) go to 340
      limit=0
      do 200 i=1,ntrn
      j=limit
      jj=0
      do 110 k=1,nbfo
  110 ib(k)=ma(k,i)
      if(i-1) 50,50,120
  115 j=j+1
      jj=0
  125 jj=jj+1
          do  10  k=1,nbfo
      ib(k)=0
      m=mb(k,j)
            l =  iabs(m)
            if  ( l .eq. 0 )   go to  10
      ib(k)=m/l*mb(l,jj)
  10        continue
c     check against previous transformations-then identity element
  120 do 30 ij=1,limit
            do  20  k=1,nbfo
      if(ib(k).ne.mb(k,ij).and.ib(k).ne.0) go to 3^
  20         continue
            go to 70
   30 continue
      do 40 k=1,nbfo
      if(ib(k).ne.k.and.ib(k).ne.0) go to 50
  40        continue
          go to 70
   50 limit=limit+1
```

```
      if(limit.le.mxtr) go to 704
      write(60,800)
      write(60,804) mxtr
      stop
  704 continue
          do  60  k=1,nbfo
   60 mb(k,limit)=ib(k)
   70 if(jj.lt.j) go to 125
      if(j.lt.limit) go to 115
      ngroup(i)=limit
  200 continue
      nrdin=ntrn
      ntrn=limit
      write(60,912) nbfo,ntrn
        do  102  i=1,nbfo
  102 write(60,910) (mb(i,j),j=1,ntrn)
c     check for zeroes in point group part of transformation matrix
      if(icon(7).ne.0) go to 7064
c     set no transformations in point group
      nhi=0
      do 7061 i=1,nrdin
      nlo=nhi+1
      nhi=ngroup(i)
      do 7062 ii=nlo,nhi
      do 7063 k=1,nbfo
      if(mb(k,ii).eq.0) go to 7065
 7063 continue
 7062 continue
 7061 continue
      nlo=nhi+1
 7065 continue
      ntrnpt=nlo-1
      go to 7066
 7064 continue
      if(ntrn.ge.ntrnpt) go to 705
      write(60,800)
      write(60,805) ntrn,ntrnpt
      stop
  705 continue
      if(ntrnpt.eq.0) go to 7066
      do 7067 i=1,nrdin
      if(ntrnpt.eq.ngroup(i)) go to 7066
 7067 continue
      write(60,2742) ntrnpt
 2742 format(//5x,'ntrnpt= ',i5,5x,'does not form a group')
      stop
 7066 continue
      write(60,578) ntrnpt
  578 format(/5x,25hno trns in point group=  ,i5)
      if(nadd.eq.0.or.ntrn.le.nadd) go to 706
      write(60,800)
      write(60,806) ntrn,nadd
      stop
  706 continue
c     ********************************************
c
c     ******************************************
c     expand tr matrix
  340 ib(1)=0
      do 250 i=2,nbfo
  250 ib(i)=ib(i-1)+ia(i-1)
      do 140 i=1,nbfn
      do 140 j=1,ntrn
  140 ma(i,j)=0
```

```
      do 15 i=1,nbfo
      jmx=ia(i)
      do 15 k=1,ntrn
      nkr=mb(i,k)
      nk=iabs(nkr)
      nnk=nkr
      if(nkr.eq.0) go to 15
      if(jmx.eq.ia(nk)) go to 707
      write(60,800)
      write(60,807) i,nk
      stop
  707 continue
      do 14 j=1,jmx
      nnbf=ib(i)+j
   14 ma(nnbf,k)=nnk/nk*(ib(nk)+j)
   15 continue
      write(60,912) nbfn,ntrn
      do 31 i=1,nbfn
      ia(i)=(i*(i-1))/2
   31 write(60,910) (ma(i,j),j=1,ntrn)
c     ********************************************
      return
  800  format(///5x,30hlabel program fatally wounded  /)
  801 format(5x,17hnclass,nbf,maxbf ,3i5)
  802 format(5x,23hnbf,sum of expand list  ,3i5)
  803 format(5x,18hntrn,ntrnpt,maxtr  ,3i5)
  804 format(5x,29htoo many trs generated-maxtr= ,3i5)
  805 format(5x,27hnot enough trs-ntrn,ntrnpt ,3i5)
  806 format(5x,18hntrn,ntr expected ,3i5)
  807 format(5x,8hclasses  ,2i3,32h have unequal no of fctns           )
  600 format(1h1//5x,26hlabel generation program          )
  610 format(//5x,20a4)
  620 format(/5x,8hoptions ,10i3)
  630 format(/5x,20hno classes of fctns= ,i3/5x,16hno basis fctns= ,i3)
  640 format(/5x,22hno symmetry elements= ,i3/5x,19hno in point group= ,
     xi3/5x,22hno elements expected= ,i3)
 905  format (24i3)
  910 format(5x,24i5)
  912 format(//5x,21htransformation matrix ,5x,4hnbf=,i4,3x,5hntrn=,i4/)
 913  format ( 10i5 )
  930 format(20a4)
      end
c
c     *********************************************
c
      subroutine olist(mtrans,list,itemp,nos,mxbf,mxtr,ntrn)
      integer*2 ii(1024),jj(1024),itg(1024)
      dimension mtrans(180,50),nos(180),list(180,50),itemp(180)
      common /ndata /  nbfn, nbfo, ndum, ntrnpt, nadd,ntape
      common/ioind/icon(10),mrec,izero,ione
      common/label/pkdlbl(1024)
      ntot=0
      nztg=0
      nxtpk=0
      iflst=0
      list(1,6)=0
      do 25 i=1,nbfn
      list(1,1)=i
      do 25 j=1,i
      list(1,2)=j
      nofild=1
          if  ( ntrn )   11,23,11
   11 ip=itemp(i)+j
          do  22  m=1,ntrn
```

```
            it =  mtrans(i,m)
            jt =  mtrans(j,m)
            iprdt =  it*jt
            itag =  1
            if  ( iprdt )   12,22,13
   12       itag =  2
   13       it =  iabs(it)
            jt =  iabs(jt)
            if  ( it - jt )   14,15,15
   14       mx =  it
            it =  jt
            jt =  mx
   15 ipt=itemp(it)+jt
      if(ip-ipt) 17,16,25
   16 if(iprdt) 25,22,22
   17 if(nofild-2) 21,18,18
   18 do 20 ic=2,nofild
      if(ipt-nos(ic)) 20,22,20
   20 continue
   21 nofild=nofild+1
      nos(nofild)=ipt
      list(nofild,1)=it
      list(nofild,2)=jt
      list(nofild,6)=itag
   22 continue
   23 nztg=nztg+1
      do 1310 m=1,nofild
      if(nxtpk-mrec) 310,316,316
  316 write(ntape) nxtpk,izero,ii,jj,itg
      ntot=ntot+nxtpk
      nxtpk=0
  310 nxtpk=nxtpk+1
      ii(nxtpk)=list(m,1)
      jj(nxtpk)=list(m,2)
      itg(nxtpk)=list(m,6)
 1310 continue
   25 continue
      write(ntape) nxtpk,ione,ii,jj,itg
      ntot=ntot+nxtpk
      write(60,456) nztg,ntot
  456 format(/5x,30hno labels-unique and total       ,2i10)
      return
      end
c
c     **********************************************
c
      subroutine tlist(mtrans,list,iiml,nos,mxbf,mxtr)
c     this routine takes all the time of label creation
      integer*2 ii(1024),jj(1024),kk(1024),ll(1024),itg(1024),muu(1024),
     1mun(1024)
      dimension mtrans(180,50),list(180,50),nos(180),iiml(180)
      common/label/pkdlbl(1024)
      common /ndata /  nbfn, nbfo, ntrn, ntrnpt, nadd,ntape
      common/ioind/icon(10),mrec,izero,ione
      data nxtpk/0/,nztg/0/,ntot/0/,ifirst/1/
c     icon(3).ne.0  used for adding basis functions to set
c     icon(5).ne.0 used for calculating integrals for an ivo calculation
      if(icon(3).gt.0) ifirst=icon(3)
      kfb=1
      if(icon(4).ne.0) kfb=ifirst
      inew=icon(5)
      list(1,6)=0
      do 38 i=ifirst,nbfn
      list(1,1)=i
```

```
      do 36 j=1,i
      list(1,2)=j
      ij2=iim1(i)+j
      ij4=(ij2*(ij2-1))/2
      kfa=1
      if(j.lt.ifirst) kfa=kfb
      do 34 k=kfa,i
      list(1,3)=k
      if(i-k) 12,11,12
   11 lmx=j
      go to 13
   12 lmx=k
   13 do 32 l=1,lmx
      list(1,4)=l
      if(inew.eq.0) go to 3636
      nnew=0
      if(i.ge.inew) nnew=nnew+1
      if(j.ge.inew) nnew=nnew+1
      if(k.ge.inew) nnew=nnew+1
      if(l.ge.inew) nnew=nnew+1
      if(nnew.gt.2) go to 32
 3636 continue
      nofild=1
      if(ntrn) 14,31,14
   14 ip=ij4+iim1(k)+1
      do 30 m=1,ntrn
      it=mtrans(i,m)
      jt=mtrans(j,m)
      kt=mtrans(k,m)
      lt=mtrans(l,m)
      iprt=it*jt*kt*lt
      itag=1
      if(iprt) 15,30,16
   15 itag=2
   16 it=iabs(it)
      jt=iabs(jt)
      kt=iabs(kt)
      lt=iabs(lt)
      if(it-jt) 17,18,18
  17            mx =  it
                it =  jt
                jt =  mx
  18            if  ( kt - lt )   19,20,20
  19            mx =  kt
                kt =  lt
                lt =  mx
  20            if  ( it - kt )   22,21,23
  21            if  ( jt - lt )   22,23,23
  22            mx =  it
                it =  kt
                kt =  mx
                mx =  jt
                jt =  lt
                lt = mx
  23            ijt =  iim1(it) + jt
                klt =  iim1(kt) + lt
      ipt=(ijt*(ijt-1))/2+klt
                if  ( ip - ipt )   25,24,32
   24 if(iprt) 32,30,30
   25 if(nofild-2) 29,26,26
   26 do 27 ic=2,nofild
      if(ipt-nos(ic)) 27,30,27
   27 continue
   29 nofild=nofild+1
```

```
      nos(nofild)=ipt
      list(nofild,1)=it
      list(nofild,2)=jt
      list(nofild,3)=kt
      list(nofild,4)=lt
      list(nofild,6)=itag
   30 continue
   31 nztg=nztg+1
      do 245 m=1,nofild
      if(nxtpk-mrec) 310,316,316
  316 write(ntape)nxtpk,izero,ii,jj,kk,ll,itg,muu
      ntot=ntot+mrec
      nxtpk=0
  310 nxtpk=nxtpk+1
      if(list(m,2)-list(m,3)) 210,220,230
  210 if(list(m,2)-list(m,4)) 211,214,217
  211 if(list(m,3)-list(m,4)) 199,212,213
  212    mun(m)=8
      go to 240
  213    mun(m)=14
      go to 240
  214 if(list(m,1)-list(m,3)) 199,215,216
  215    mun(m)=2
      go to 240
  216    mun(m)=11
      go to 240
  217 if(list(m,1)-list(m,3)) 199,218,219
  218    mun(m)=10
      go to 240
  219    mun(m)=13
      go to 240
  220 if(list(m,3)-list(m,4)) 199,221,224
  221 if(list(m,1)-list(m,2)) 199,222,223
  222    mun(m)=1
      go to 240
  223    mun(m)=6
      go to 240
  224 if(list(m,1)-list(m,2)) 199,225,226
  225    mun(m)=4
      go to 240
  226    mun(m)=9
      go to 240
  230 if(list(m,1)-list(m,2)) 199,231,234
  231 if(list(m,3)-list(m,4)) 199,232,233
  232    mun(m)=3
      go to 240
  233    mun(m)=5
      go to 240
  234 if(list(m,3)-list(m,4)) 199,235,236
  235    mun(m)=7
      go to 240
  236    mun(m)=12
  240 continue
      ii(nxtpk)=list(m,1)
      jj(nxtpk)=list(m,2)
      kk(nxtpk)=list(m,3)
      ll(nxtpk)=list(m,4)
      itg(nxtpk)=list(m,6)
      muu(nxtpk)=mun(m)
  245 continue
   32 continue
   34 continue
   36 continue
   38 continue
```

```
      ntot=ntot+nxtpk
      write(ntape)nxtpk,ione,ii,jj,kk,ll,itg,muu
      write(60,456) nztg,ntot
  456 format(/5x,30hno labels-unique and total       ,2i10)
      return
  199 write(60,145) (list(m,ku),ku=1,4)
  145 format(/5x,'error in mlist -i,j,k,l ',4i5)
      stop
      end
```

APPENDIX D
LISTING OF THE MAIN LOPAS PROGRAM
PARALLEL PROCESSOR VERSION

A. B. Kunz, Author

deserver:barry

parlop.f

Tue Feb 26 10:37:48 1991

lw / TCD LaserWriter II NT

```
lw deserver:barry  Job: parlop.f  Date: Tue Feb 26 10:37:48 1991

lw deserver:barry  Job: parlop.f  Date: Tue Feb 26 10:37:48 1991

lw deserver:barry  Job: parlop.f  Date: Tue Feb 26 10:37:48 1991

lw deserver:barry  Job: parlop.f  Date: Tue Feb 26 10:37:48 1991
```

```
C     This is an implementation
C     of local orbitals procedures of
C     Adams, Gilbert and Kunz implemented
C     for cluster building blocks and
C     a gaussian basis set
C     part of the MEGAMOL sequence
C     Molecules for the 90's
C     author is A B Kunz
C     Michigan Technological University
C     College of Engineering
C     Fortran 77
C     written 1991
C     written for parallel computers using the cosmic environment
C     all rights reserved by the author
C
C     ***************************************************************
C
      Program lopas
C
C     ***************************************************************
C
      implicit real*8(a-h,o-z)
      dimension nenv(20),id(20,200),
     1trans(85),xof(20,200),yof(20,200),zof(20,200),
     2a(20,200),b(20,200),c(20,200)
      real*4 tyme(2)
      real*8 norm
      common/angle/angl(73),cangl(73),sangl(73)
      common/pparms/all,any,iid,ipid,isiz,izro
      character*11 mol41(20)
      integer all,any
C
C     ***************************************************************
C
    1 format(i4)
    2 format(' THIS IS A GAUSSIAN BASIS SET LOPAS CALCULATION ',/,
     1' USING THE MULTI CENTER METHOD OF A B KUNZ ',/,
     2' FOLLOWING THE PROCEDURE OF ADAMS-GILBERT-KUNZ ')
    3 format(1x,' nbb = ',i4)
    4 format(i4)
    5 format(i4,6x,6f10.4)
    6 format(1x,' nenv(i) = ',i4)
    7 format(1x,' id = ',i4,' xof = ',f10.4,' yof = ',f10.4,
     1' zof = ',f10.4,/,'  angle 1 = ',f10.4,' angle 2 = ',f10.4,
     2' angle 3 = ',f10.4)
    8 format (1x,' CPU run time is ',f16.3,' sec ')
C
C     ***************************************************************
C
      mol41(1)='mol4101.dat'
      mol41(2)='mol4102.dat'
      mol41(3)='mol4103.dat'
      mol41(4)='mol4104.dat'
      mol41(5)='mol4105.dat'
      mol41(6)='mol4106.dat'
      mol41(7)='mol4107.dat'
      mol41(8)='mol4108.dat'
      mol41(9)='mol4109.dat'
      mol41(10)='mol4110.dat'
      mol41(11)='mol4111.dat'
      mol41(12)='mol4112.dat'
      mol41(13)='mol4113.dat'
      mol41(14)='mol4114.dat'
      mol41(15)='mol4115.dat'
      mol41(16)='mol4116.dat'
      mol41(17)='mol4117.dat'
```

```
      mol41(18)='mol4118.dat'
      mol41(19)='mol4119.dat'
      mol41(20)='mol4120.dat'
      iid=mynode()
      isiz=nnodes()
      ipid=mypid()
      all=-1
      any=-2
      izro=0
c
c     ****************************************************************
c
      if(iid.eq.0)then
c
c     ****************************************************************
c
      open(unit=60,file='mol5e.dat',form='formatted')
      open(unit=14,file='mol14.dat',form='formatted')
c
c     ****************************************************************
c
c     define local orbitals problem
c
      read(14,1)nbb
      write(60,2)
      write(60,3)nbb
      print 2
      print 3,nbb
      do 20 i=1,nbb
      read(14,4)nenv(i)
      do 20 j=1,nenv (i)
      read(14,5)id(i,j),xof(i,j),yof(i,j),zof(i,j),a(i,j),b(i,j),c(i,j)
   20 continue
      do 21 i=1,nbb
      write(60,6)nenv(i)
      print 6,nenv(i)
      do 21 j=1,nenv(i)
      write(60,7)id(i,j),xof(i,j),yof(i,j),zof(i,j),a(i,j),b(i,j),c(i,j)
   21 print 7,id(i,j),xof(i,j),yof(i,j),zof(i,j),a(i,j),b(i,j),c(i,j)
c
c     lopas set up data done now
c     Do local orbital buildingblocks in free space now
c
c     ****************************************************************
c
c     broadcast nbb,nenv,id,xof,yof,zof,a,b,c here
      do 60 ii=1,isiz-1
      nm=100*ii+1
      l=4
      call frecv(ij,l,nm,ii,ipid)
      if(ij.ne.ii)stop ' par punt '
      nm=nm+1
      l=4
      call fsend(nbb,l,nm,ii,ipid)
      nm=nm+1
      l=80
      call fsend(nenv,l,nm,ii,ipid)
      nm=nm+1
      l=16000
      call fsend(id,l,nm,ii,ipid)
      nm=nm+1
      l=32000
      call fsend(xof,l,nm,ii,ipid)
      nm=nm+1
      call fsend(yof,l,nm,ii,ipid)
      nm=nm+1
```

263

```
      call fsend(zof,l,nm,ii,ipid)
      nm=nm+1
      call fsend(a,l,nm,ii,ipid)
      nm=nm+1
      call fsend(b,l,nm,ii,ipid)
      nm=nm+1
      call fsend(c,l,nm,ii,ipid)
   60 continue
      else
      nm=iid*100+1
      l=4
      call fsend(iid,l,nm,izro,ipid)
      nm=nm+1
      l=4
      call frecv(nbb,l,nm,izro,ipid)
      nm=nm+1
      l=80
      call frecv(nenv,l,nm,izro,ipid)
      nm=nm+1
      l=16000
      call frecv(id,l,nm,izro,ipid)
      nm=nm+1
      l=32000
      call frecv(xof,l,nm,izro,ipid)
      nm=nm+1
      call frecv(yof,l,nm,izro,ipid)
      nm=nm+1
      call frecv(zof,l,nm,izro,ipid)
      nm=nm+1
      call frecv(a,l,nm,izro,ipid)
      nm=nm+1
      call frecv(b,l,nm,izro,ipid)
      nm=nm+1
      call frecv(c,l,nm,izro,ipid)
      endif
      ilop=0
      do 30 i=iid+1,nbb,isiz
c
c     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
c
c*****this is a par-do
c
c     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
c
      call lister(i,ilop)
      call poly(i,ilop)
      call uhf(i,ilop)
   30 continue
      if(iid.ne.0)then
      nm=200*iid
      l=4
      call fsend(iid,l,nm,izro,ipid)
      else
      do 70 ii=1,isiz-1
      nm=ii*200
      l=4
      call frecv(ij,l,nm,ii,ipid)
      if(ij.ne.ii) stop ' par punt two '
      call fkill(ii,ipid)
   70 continue
      tymes=etime(tyme)
      print 8,tyme(1)
      write(60,8)tyme(1)
      if(ilop.eq.0) stop ' done for now '
c
c     ****************************************************************
```

```
c
c     free space estimates of building blocks are
c     evaluated here
c     get multipole moments and begin lopas rotations
c
c     *****************************************************************
c
      do 9999 ilps=1,4
      ilop=ilps
c     evaluate moments of each lopas block here
c     evaluate detailed potentials as well
      do 40 i=iid+1,nbb,isiz
c
c     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
c
c*****this is a par-do
c
c     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
c
   40 call moments(i,ilps)
c
c     *****************************************************************
c
c     broadcast needed moments data to each other processor now
c
c     *****************************************************************
c
      do 80 i=1,nbb
      open(unit=41,file=mol41(i),form='unformatted')
c     do I send or do I receive data?
c     if  i = iid  I will send
c     otherwise I will receive data
      if(i.ne.iid) go to 81
c     I will broadcast data
      read(41)trans
      l=4
      nm=80
      do 82 ii=1,nbb
      if(i.eq.ii) go to 82
      call frecv(ij,l,nm,ii,ipid)
      if(ij.ne.ii)stop ' transfer tilt after moments '
      l=680
      nm=81
      call fsend(trans,l,nm,ii,ipid)
   82 continue
c     all data sending is complete
      go to 80
   81 continue
c     receive data and store it
      l=4
      nm=80
      call fsend(iid,l,nm,i,ipid)
      l=680
      nm=81
      call frecv(trans,l,nm,i,ipid)
      write(41)trans
   80 close(unit=41)
c
c     *****************************************************************
c
c     data exchange complete
c
c     *****************************************************************
c
      print 19
c     moments and w00 potential formed for each block
```

```
C     form potential of environment of each building block
      do 50 i=iid+1,nbb,isiz
C
C     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C
C*****this is a par-do
C
C     $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C
      call pot(ilps,nenv,xof,yof,zof,a,b,c,i,id)
      print 18
      call uhf(i,ilop)
   50 continue
 9999 continue
      close(unit=61)
      stop 'lopas complete'
      end
      Subroutine moments(ibb,ilpss)
c     calculates V00 potential for this building block
c     calculates moments as well
c     uses spherical coordinates and 3-d numerical quadrature
c     authored by A B Kunz
c     Fortran 77
c     all rights reserved by the author
c     integrations use Weddle's rule over angles
c     integrations use Simpson's rule over r
c     weighting factors are used to define each
c     specific integrations properties and as an
c     aid to high speed computation. This reduces each
c     integral to a dot product, and facilitates vectorization
c     by a smart compiler
c
c***********************************************************************
c
      implicit real*8(a-h,o-z)
      dimension rho(81,37,73),r(81),ri1(37,73),ri2(73),wr(81),
     1wa1(37),wa2(73),ntype(180),nfirst(180),nlast(180),
     2fodm(180,180),t1(218781),t2(218781),t3(218781),nr(20
     3,3),t5(81,37,73),t4(81,37,73),eta(1024,5),c(1024)
     4,ps(360),v(81),v1(81),v2(81)
      common/angle/angl(73),cangl(73),sangl(73)
      common/mompot/vlist(1024,4),ntype,nfirst,nlast,eta,c
      real*8 norm
      real*4 ain(180),psi(2,180,180)
      character*4 zl(20)
      character*15 mol51(20)
      character*11 mol5a(20),mol11(20),mol04(20),mol30(20),mol41(20),
     1mol40(20)
      equivalence(t3(1),rho(1,1,1))
      equivalence(t4(1,1,1),t1(1))
      equivalence(t5(1,1,1),t2(1))
      data  nr /  0,1,0,0,2,0,0,1,1,0,3,0,0,2,2,1,0,1,0,1,
     x            0,0,1,0,0,2,0,1,0,1,0,3,0,1,0,2,2,0,1,1,
     x            0,0,0,1,0,0,2,0,1,1,0,0,3,0,1,0,1,2,2,1 /
c
c***********************************************************************
c
      mol51(1)='/work/psi01.dat'
      mol51(2)='/work/psi02.dat'
      mol51(3)='/work/psi03.dat'
      mol51(4)='/work/psi04.dat'
      mol51(5)='/work/psi05.dat'
      mol51(6)='/work/psi06.dat'
      mol51(7)='/work/psi07.dat'
      mol51(8)='/work/psi08.dat'
      mol51(9)='/work/psi09.dat'
```